University of California
Santa Barbara

Sequential Design for Gaussian Process Surrogates in Noisy Level Set Estimation

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Statistics and Applied Probability

by

Xiong Lyu

Committee in charge:

Professor Mike Ludkovski, Chair
Professor Yuedong Wang
Professor Sang-Yun Oh

March 2020

The Dissertation of Xiong Lyu is approved.

Professor Yuedong Wang

Professor Sang-Yun Oh

Professor Mike Ludkovski, Committee Chair

March 2020

Sequential Design for Gaussian Process Surrogates in Noisy Level Set Estimation

by

Xiong Lyu

For grandfather, in memoriam.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor Professor Mike Ludkovski for the continuous support of my Ph.D. study and research, and for his patience, motivation, enthusiasm and immense knowledge. He continually and convincingly conveyed a spirit of adventure with regard to research, and an excitement with regard to teaching, both of which always motivate me to keep challenging and working hard in research and career. His guidance helped me in all the time of research and writing of this thesis. Without his persistent help, this dissertation would not have been possible.

I would also like to thank Professor Wang, who introduced the advanced statistical methods to me, and was always patient for discussion and advices; and thank Professor Oh, who firstly introduced machine learning and data mining to me, which turned out to be solid background for my internships and future career. I really appreciate for them serving as members of my committee, and for their encouragement, insightful comments and hard questions.

In addition, I am grateful to all professors in the department and professors who visited the department in the past five years for providing great statistical courses and inspiring insights on various research topics. I also would like to thank all fellows and friends at UC Santa Barbara, who were always there supporting me over the five years. They are significant parts of my enjoyable and precious memory here.

Last but not least, I wish to thank my parents and my grandparents, who were always there to believe what I believe in, to support what I pursue with and to encourage me when I have a hard time, and my husband, who kept challenging me and always gave me "bittersweet" support for research and life. This dissertation is dedicated to my grandfather, who passed away during my Ph.D. time. He inspired my curiosity to the world and will always be there watching me to accomplish my dreams.

Curriculum Vitæ

Xiong Lyu

Education

2019	Ph.D. in Statistics and Applied Probability, University of California, Santa Barbara.
2016	M.A. in Statistics and Applied Probability, University of California, Santa Barbara.
2014	B.S. in Mathematics, Zhejiang University, Hangzhou, China.

Publications

Lyu, X., Binois, M. and Ludkovski, M., 2018. Evaluating gaussian process metamodels and sequential designs for noisy level set estimation. arXiv preprint arXiv:1807.06712.

Lyu, X., and Ludkovski, M., 2019. Adaptive batching for Gaussian Process surrogates with application in noisy level set estimation. *Working in Progress*

Conferences

Dec 2017	Thirty-first Conference on Neural Information Processing Systems, Long Beach, CA
Mar 2019	Amazon's Graduate Research Symposium, Seattle, WA
Oct 2019	Women in Statistics and Data Science, Bellevue, WA

Abstract

Sequential Design for Gaussian Process Surrogates in Noisy Level Set Estimation

by

Xiong Lyu

We consider the problem of learning the level set for which a noisy black-box function exceeds a given threshold. To efficiently reconstruct the level set, we investigate Gaussian process (GP) metamodels and sequential design frameworks. Our focus is on strongly stochastic samplers, in particular with heavy-tailed simulation noise and low signal-to-noise ratio. We introduce the use of four GP-based metamodels in level set estimation that are robust to noise misspecification, and evaluate the performance of them. In conjunction with these metamodels, we develop several acquisition functions for guiding the sequential experimental designs, extending existing stepwise uncertainty reduction criteria to the stochastic contour-finding context. This also motivates our development of (approximate) updating formulas to efficiently compute such acquisition functions for the proposed metamodels. To expedite sequential design in stochastic experiments, we also develop adaptive batching designs, which are natural extensions of sequential design heuristics with the benefit of replication growing as response features are learned, inputs concentrate, and the metamodeling overhead rises. We develop four novel schemes that simultaneously or sequentially determine the sequential design inputs and the respective number of replicates. Our schemes are benchmarked by using synthetic examples and an application in quantitative finance (Bermudan option pricing).

Contents

Curriculum Vitae **vi**

Abstract **vii**

1 Introduction **1**
1.1 Toy Example 2
1.2 Structure of Thesis 5

2 Background and Related Work **7**
2.1 Statement of Problem 7
2.2 Motivation 10
2.3 Related Work 13
2.3.1 Gaussian Process 13
2.3.2 One-stage Design of Experiments 14
2.3.3 Sequential Design 15
2.3.4 Efficient Sequential Designs 19
2.4 Summary of Contributions 23

3 Level Set Estimation with Gaussian Process Surrogates **28**
3.1 Gaussian Process Regression with Gaussian Noise 28
3.2 Gaussian Process Regression with Student t-Noise 30
3.3 Gaussian Process Classification 32
3.4 Gaussian Process with Monotonicity Constraint 34
3.5 Student-t Process Regression with Student-t Noise 36
3.6 Metamodel Performance for Level Set Inference 37

4 Sequential Design for Gaussian Process Surrogates **43**
4.1 Sequential Design 43
4.1.1 Literature Overview 44
4.1.2 Acquisition Functions for Level Set Estimation 48
4.2 Look-Ahead Variance 57

4.2.1 Computation Details for Look-Ahead Variance . . . 61
4.3 Synthetic Experiments . . . 65
4.3.1 Benchmark Construction . . . 65
4.3.2 Comparison of GP Metamodels . . . 69
4.3.3 Empirical Errors and Uncertainty Quantification . . . 71
4.3.4 Designs for Contour Finding . . . 73
4.4 Application to Optimal Stopping Problems in Finance . . . 74
4.4.1 Results . . . 77

5 Adaptive Batching for Gaussian Process Surrogates 88
5.1 Gaussian Processes with Batched Inputs . . . 88
5.2 Adaptive Designs . . . 90
5.2.1 Level Set Estimation . . . 90
5.2.2 Fixed Batch Design . . . 90
5.2.3 Multi-Level Batching . . . 91
5.2.4 Ratchet Batching . . . 92
5.2.5 Adaptively Batched Stepwise Uncertainty Reduction . . . 93
5.3 Adaptive Design with Stepwise Allocation . . . 95
5.3.1 Allocation Rule for GP . . . 99
5.3.2 Allocation Rule for t-GP . . . 101
5.4 Synthetic Experiments and Case Study . . . 104
5.4.1 Synthetic Experiments and Computational Implementation Details . . . 104
5.4.2 Algorithm Performance . . . 106
5.4.2.1 Comparing Designs . . . 111
5.4.3 Application to Optimal Stopping . . . 115

6 Conclusions and Future Works 121

Bibliography 125

Chapter 1

Introduction

Simulation has become widespread for approximating or analyzing dynamic, stochastic proposed or existing systems. Virtually any level of detail can be modeled and any performance measure estimated, which explains simulation's popularity [1]. They are commonly used in environmental (Gotovos et al. [38]), social (Cioffi-Revilla [27]) and biological (Johnson [49]) sciences, where the design of a system has to take into account the fact that some design parameters are subject to unknown variations that may affect the reliability/performance of the system. Specifically, the components that affect the performance of the system are named as design variables or inputs, and the response is the latent performance function which is derived from the observations from a stochastic model of the system. For example, in order to find the optimal stopping criteria for Bermudan option, financial engineers simulate the future payoff as a function of stock prices. The future payoff is the design objective of interest, or namely, the response, and the stock prices are the inputs. The observed simulation output is *observation*. Noisier simulations demand substantial experiments to isolate the signal from noise. Therefore, large amount of simulations might be required to construct a complicated simulation model and desirable results may not be produced in a timely manner. Decision of using a simulation model for a large-scale and complex system lften represents a significant investment of time and money.

In a number of applications where computer experiments are used to characterize the behavior of a parametric system of interest, scientists not only wish to predict the response for a given input, but are interested in recovering a set of inputs leading to a given range of values for the response. Rather than aiming to capture the precise shape of the response over the entire domain, in this research we are interested in estimating the *level set* where the response exceeds some particular threshold.

In real life, it is not always practical to construct prototype versions of the system due to the expense that prohibits large-scale experimentation. One common solution is to construct a metamodel (or equivalently surrogate) that is built on observations from simulations run at several selected inputs and models the response as a function over the entire research domain. Metamodels are approximation models that mimic the behavior of the simulation model as closely as possible while being computationally cheaper to evaluate. The exact working of simulation codes is assumed to be unknown, and only the input-output behavior is important. Consequently this approach is also named as *black-box* modeling. These techniques are widely used in the computer experiments and machine learning, see e.g. Jones et al. [51], Santner et al. [93], Dubourg et al. [30] and Bect et al. [6].

1.1 Toy Example

Figure 1.1 provides an illustration of how the metamodels work. Our target is to estimate the contour between two level sets, one with positive response and the other with negative response. This contour is also named as zero-contour. The observations are simulated from a 2-D Branin-Hoo function with Gaussian distributed noise (detailed set-up of this experiment is described in Section 5.4). Estimation of the level sets consists of two steps: (1) build a metamodel based on observations from simulations run at the inputs; (2) select the inputs sequentially via optimizing a statistical rule, or namely acquisition function, which measures the uncertainty of the current

estimate. For step (1), the Gaussian Process/kriging methodology is popular in the design and analysis of deterministic computer experiments (DACE). Ankenman [1] extended Gaussian Process to stochastic simulations. Step (2) is also known as *sequential design*. As mentioned in the beginning of this thesis, it may involve complex and time-consuming simulations to obtain the observations, which turns the response into an expensive-to-evaluate function. Therefore, the estimation of level sets must be carried out with a restricted number of observations, generally excluding a Monte Carlo approach. Sequential design strategies are highly successful for selecing the restricted number of observations.

In our implementation, the initialization is done via a space-filling Latin hypercube design [70] (sampling equally across the entire input space). The metamodel based on the initial twenty inputs provides a raw estimate of the true response, and the estimated contour is far from the zero-contour (shown in the upper right panel). With Gaussian Process as the metamodel (alternative metamodels are discussed in Chaper 3), we have an estimate of the approximation uncertainty. So we can define an acquisition function (Chapter 4 provides an overview of existing acquisition functions and proposes four new heuristics for level set estimation) to guide input argumentation by iteration. Basically in each iteration, we choose the input which optimizes the acquisition function (usually it is close to the estimated zero-contour), simulate one observation (or multiple observations for batching design, to be covered in Chapter 5) at the chosen input, and add the input with its observation into the currect dataset. Then a new estimate of the zero-contour is obtained based on the new dataset and the acquisition function is updated. We iterate this step ten rounds and add ten new inputs, especially close to the zero-contour, the estimate is more accurate (shown in the lower left panel). After another 70 iterations, the estimate is close to the zero-contour, with a narrower confidence band, indicating that there is less approximation uncertainty.

Our analysis is driven by the primal effect of noise on level set estimation. This effect was already documented in related studies, such as that of Jalali et al. [48] who observed the

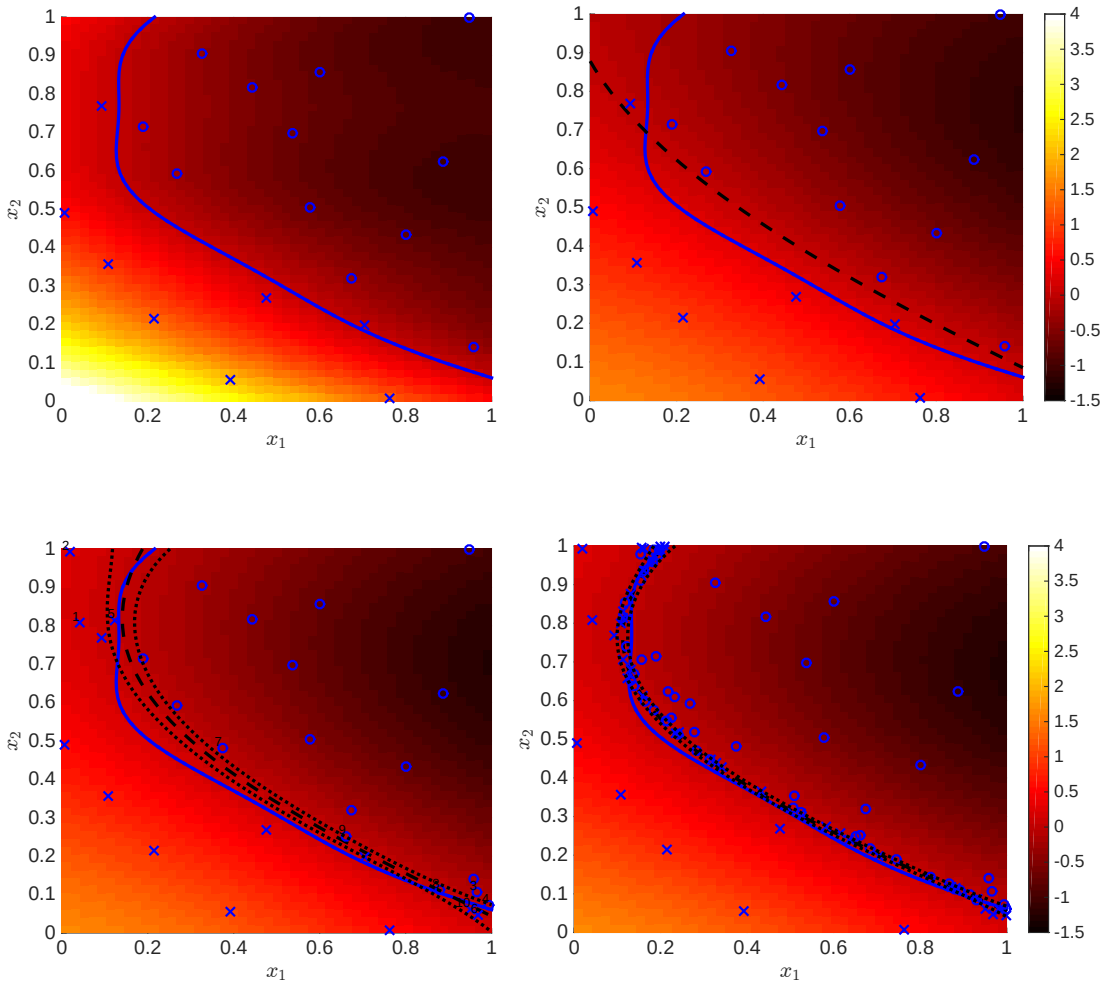


Figure 1.1: Illustration of level set estimation. Upper left: true surface, with 20 observations (blue cross for negative observations, blue circle for positive observations). Blue solid line represents the zero-contour between the positive and negative response. Shade of background represents the value of response. Upper right: estimated zero-contour (black dashed line) by a surrogate based on the initial observations. Lower left: estimated zero-contour (black dashed line) with its 95% confidence band (black dotted line) after adding sequentially ten new observations. Labels respresent the index of iterations. Lower Right: estimated zero-contour (black dashed line) with its 95% confidence band (black dotted line) after adding sequentially eighty new observations.

strong impact of noise on performance of *Bayesian Optimization* (a topic closely related to level set estimation, details to be mentioned in Chapter 2). Consequently, specialized metamodel frameworks and sequential design criteria to argument input set are needed that can best handle

the stochasticity for the given loss specification. Thus, we focus on studying the combination of the sequential design criteria with the metamodel frameworks that aims to strike the best balance in carrying out uncertainty quantification and constructing a robust surrogate that is not too swayed by the simulation noise structure.

Another difficulty is how to increase the simulation size N while still keeping the metamodeling manageable. Although more robust metamodeling frameworks relieve the burden of stochasticity in noisy level set estimation, their performance is greatly limited by the simulation size N. While increasing N helps to relieve the tension of low signal-to-noise ratio observations, challenges are posed on both computation side and memory side as the value of N explodes. For example. the computational complexity for a common metamodel, Gaussian Process (GP), is $\mathcal{O}(N^3)$, and this cubic computation severely limits the simulation size N that can be modeled with GPs.

1.2 Structure of Thesis

Motivated by study cases in Bermudan option, the contributions of this thesis focus on both metamodel frameworks for noisy level set estimation and adaptive batching designs. The rest of this thesis is organized as follows.

Chapters 1 and 2 introduce the general context and related works, starting with this Chapter 1. Chapter 2 firstly describes the statement of problem, and then focuses on several key related design frameworks. We also summarize the main contributions of this dissertation in the last section of Chapter 2.

Chapter 3 describes the Gaussian Process metamodels we employ for stochastic computer experiments. Besides Gaussian Process with Gaussian observations, we extended the models to Gaussian Process with Student-t observations where we assume the noise follows Student-t distribution, classification Gaussian Process where the response is the sign of response,

monotonic Gaussian Processes where we assume the response is monotone, and Student-t Process where we assume the response follows a Student-t process. The Student-t Process section is co-authored with Mickaël Binois.

In Chapter 4, we develop the sequential designs for the metamodels for the level set estimation problem, and discuss the look-ahead variance formulas for non-Gaussian GPs. We compare the models using synthetic data where ground truth is known. Two case studies from derivative pricing are investigated to further compare the proposed metamodels. Chapters 3 and 4 integrate into the following paper:

- Lyu, X., Binois, M. and Ludkovski, M., 2018. Evaluating gaussian process metamodels and sequential designs for noisy level set estimation. arXiv preprint arXiv:1807.06712.

In Chapter 5, we develop adaptive batching heuristics for sequential designs that jointly optimize over the new input and replication level. We also take a different track and explore dynamic replication through allocating new simulations to existing inputs. We benchmark the proposed schemes on three synthetic case studies and examples from Bermudan option pricing. Chapter 5 corresponds to the following paper:

- Lyu, X., and Ludkovski, M., 2019. Adaptive batching for Gaussian Process surrogates with application in noisy level set estimation. *Working in Progress*

In Chapter 6, we conclude the thesis and propose potential directions for future research.

Our algorithms are implemented in MATLAB (R2014b or newer versions) and incorporated in the public package *GPstuff* (by Vanhatalo et al. [105]), which is a versatile collection of Gaussian Process metamodels and computational tools required for inference. Implementation details of synthetic experiments and case studies in this thesis are also included. The full package can be found in Github

- `https://github.com/GPBatch/Gaussian-Process-with-Batch-Design`.

Chapter 2

Background and Related Work

2.1 Statement of Problem

Level set estimation is common in reliability engineering, which focuses on describing the set of parameter configurations leading to an unsafe design (see Dubourg et al. [30] for reliability measurement, and Picheny et al. [83] and Bect et al. [6] for measuring the probability of failure). In both cases, reliability is typically measured by a failure probability, determined via the limit state of a performance function [81], and the main focus is to ensure that the latter is lower than a given threshold so that the system is safe. The choice of this threshold depends on the stakeholders or the system to be designed. For example, in the field of part tolerance testing a commonly admitted unit for assessing the non-conformity rates is the number of rejected parts per million (ppm) [30]. Other applications include works from Wang et al. [107] for recovering parameters of robotic control system, with the ratio of liquid ended up in the target cup poured by a robot as the response and the robot configurations at the input, as well in natural sciences, where conditions leading to dangerous phenomena in climatological [5] or environmental [38] or geophysical [35] settings are of crucial interest. In the latter fields that directly impact human lives (such as the nuclear industry or climate engineering), the selection

of an acceptable probability of dangerous phenomena is much more serious and beyond the scope of this thesis. It also arises intrinsically in control frameworks where one wishes to rank the pay-off or performance from several available actions [44].

We consider a setup where the response is modeled by a continuous latent function $f : D \to \mathbb{R}$ over a d-dimensional domain $D \subseteq \mathbb{R}^d$. In stochastic simulations, for any $x \in D$, we have access to a simulator $Y(x)$ that generates noisy samples of $f(x)$:

$$Y(x) = f(x) + \epsilon(x), \tag{2.1}$$

where $\epsilon(x)$ are realizations of independent, mean zero random variables with variance $\tau^2(x)$. Besides observation noise as with in real life, numerical simulations may also suffer from other source of noise. For example, if running the same simulation twice, observations obtained are not the same with codes based on Monte Carlo methods. The observation depends on the number of runs or the size of meshes. Also, some physical conditions are intrinsically unstable, especially in high speed simulations, where the observation is generated differently depending on numerical noise (due to e.g. the number of cores used or architecture of high performance computers) [9].

The level set estimation problem consists in classifying every input $x \in D = S \cup N$ according to

$$S = \{x \in D : f(x) \geq h\}, \qquad N = \{x \in D : f(x) < h\}. \tag{2.2}$$

S and N are the level sets, which are sets of inputs based on whether the level of its response f is greater or smaller than a threshold h. Without loss of generality the threshold h is taken to be zero, so that the level set estimation is equivalent to learning the sign of the response f. For later use we also define the corresponding zero-contour of f, namely the partition boundary (or

zero-contour) $\partial S = \partial N = \{x \in D : f(x) = 0\}$. These two topics are closely related: they all require estimate of the sign of response f. Therefore, any strategies proposed for the second objective are expected to perform reasonably well on the first one, which is the goal of this thesis.

Two other topics are closely related to level set estimation. One is *bandit optimization*, where we need to sequentially evaluate a noisy black-box function $f : D \to \mathbb{R}$ over an input space $D \subseteq \mathbb{R}^d$ with the goal of finding its optimum: $x^* = \arg\min_{x \in D} f(x)$. Applications include hyper-parameter tuning in machine learning [46, 96], scientific experiments [37, 80] and optimal policy search [62, 69]. In above applications, the response is the loss of a (machine learning/experimental) system, and the inputs are the (hyper-) parameters/configurations of the system. Similar to level set estimation, typically the evaluation is expensive and the related literature focuses on developing methods for finding the optimum while keeping the number of evaluations to f at a minimum. The optimization problem can be viewed as a special case of level set estimation, since our region of interest is reduced from the zero-contour for level set estimation into one optimal point for optimization. Although this research mainly focuses on level set estimation, most ideas can be extended to solve optimization problems as well.

The other relevant topic is surface metamodeling of the response, where our concern is the accuracy of metamodel over the entire domain. In comparison, for level set estimation, our concern is the weighted accuracy of the metamodel, with more weight on region close to the zero-contour and less otherwise. For optimization problem, we target on learning the optimal input accurately. Applications of surface metamodeling include urban transportation analysis [78], with the vehicle travel time as the response and the green times for the signalized lanes as the inputs. For surface metamodeling problems, our goal is to minimize the integrated mean square error (IMSE) between the true response and the estimated response over the entire input space.

To assess a level-set estimation algorithm, we compare the resulting estimate $\hat{S}$ with the true

S in terms of their symmetric difference. Let μ be a probability measure on the Borel σ-algebra $\boldsymbol{\mathcal{B}}(D)$ (e.g., $\mu = \text{Leb}_D$). Then our loss function is

$$L(S, \hat{S}) = \mu(S \Delta \hat{S}) \qquad \text{where} \quad S_1 \Delta S_2 := (S_1 \cap S_2^C) \bigcup (S_1^C \cap S_2). \tag{2.3}$$

Frequently, the inference is carried out by first producing an estimate $\hat{f}$ of the latent function f; in that case we take

$$\hat{S} = \{x \in D : \hat{f}(x) \geq 0\}, \tag{2.4}$$

and rewrite the loss as

$$L(f, \hat{f}) = \int_{x \in D} \mathbb{I}(\text{sign}\, \hat{f}(x) \neq \text{sign}\, f(x)) \mu(dx), \tag{2.5}$$

where $\mathbb{I}(\cdot)$ is the indicator function.

2.2 Motivation

A concrete example of level set estimation, which motivated this thesis, comes from simulation-based algorithms for valuation of Bermudan options [39, 64]. This problem consists of maximizing the expected reward $h(\tau, X_\tau)$ over all stopping times $\tau \in \{0, \Delta t, 2\Delta t, \ldots, T\}$ bounded by the specified horizon T:

$$V(t, x) = \sup_{\tau \geq t, \tau \in \mathcal{S}} \mathbb{E}[h(\tau, X_\tau) | X_t = x], \tag{2.6}$$

where (X_t) is the underlying asset price at time t, typically satisfying a stochastic differential equation and Δt is the frequency of exercising. The approach in the so-called Regression Monte

Carlo methods [63, 102] is to convert the decision of whether to exercise the option

$$\tau(t, x) = t,$$

or continue

$$\tau(t, x) > t,$$

when $X_t = x$ at intermediate step t, into comparing the immediate reward $h(t, x)$ vis à vis the reward-to-go $C(t, x)$. In turn this is equivalent to determining the level set (known as the continuation region)

$$S_t = \{x \in D : f(x; t) \geq 0\},$$

based on the timing value

$$f(x; t) := C(t, x) - h(t, x).$$

The stopping problem (2.6) is now solved recursively by backward induction over $t = T - \Delta t, T-2\Delta t, \ldots$, which allows noisy samples of $f(x; t)$ to be generated by simulating a trajectory $X^x_{t:T}$ emanating from x and evaluating the respective *pathwise* reward-to-go. Probabilistically, this means that we are interested in (2.1) where f corresponds to a *conditional expectation* related to a path-dependent functional of the Markov process $X_\cdot$; the average of realized payoffs along a large amount of paths arises naturally as a metric regarding the quality of the estimated stopping rule in terms of the underlying distribution $\mu(\cdot; t)$ of X_t.

Regression Monte Carlo (RMC) [63] has emerged as the most popular method to solve optimal stopping problem for American/Bermudan options due to its flexible and simple implementation, as well as its strong empirical performance [39]. Longstaff and Schwartz [63] proposed computing the Snell envelope by combining iterative Dynamic Programming (DP) with a Monte Carlo approximation to (2.6). The key innovation of their work over the earlier literature (see [19]) is that the (2.6) was firstly used to guide decision making process, instead

of quantification of the expected value. It also motivates us to link the optimal stopping criteria for Bermudan options problem to level set estimation. However, concerns about computational space and time arise in high-dimensional, complex models. One variety of solutions focuses on simulation-based methods beyond RMC, including quantization techniques [4], Malliavin calculus methods [12], stochastic mesh schemes [15] and sequential design frameworks [39, 64]. We refer to [64] for a summary of existing state of the art and the connection to employing a GP metamodel for learning the timing value $f(\cdot; t)$.

There are several common challenges among the motivated examples and related applications: (1) The true response or underlying function is unknown. For example, there is no true function to "map" the stock price to the future payoff in the Bermudan option example. In these problems, we are in a *black-box* setting. As a result, there is no analytical solution to estimate the zero-contour in level set estimation. Also, looking for the optimum by applying the gradient descent technique in bandit optimization is inconvenient or even infeasible. (2) It is expensive to obtain observations. It may take a lot of time for each simulation, or it may be cheap to obtain one simulation, but we need a large amount of simulations, leading to exploding cost. Bandit optimization for hyperparameters in complex machine learning system is one example for the former case, where we may need hours or even days to obtain one simulation. For the latter case, in Bermudan option example, we need to simulate the entire path for stock price movement to obtain one simulated future payoff. Although the time for one simulation is trivial, usually tens of thousands of simulations are required and thus the simulation time is expensive. Or it may also take a lot of effort/expense to obtain each observation. Examples include quantifying configuration variables in automatic driving. Consequently, the inference is based on a small set of carefully chosen inputs that are determined one by one via a Bayesian procedure (sequential design); (3) The observations are noisy. The observations may have low signal to noise ratio, or the simulations may be heteroscedastic, for example, the assemble-to-order (ATO) problem introduced by Hong and Nelson [43] and the epidemics management problem discussed by

Ludkovski and Niemi [65]. Besides, there is an inner tension between (2) and (3): to handle mis-specified noise or low signal to noise ratio, we need a large amount of observations to learn the region of interest. However, (2) limits the total number of observations we could use for manageable computation. How to strike balance between (2) and (3) is also one area this thesis focuses on.

2.3 Related Work

Reconstructing S via a metamodel can be divided into two steps: the construction of the response model and the development of methods for efficiently selecting the simulation inputs $x_{1:N}$, known as design of experiments (DoE). Since the level set is intrinsically defined in terms of the unknown f, an *adaptive* DoE approach is needed that selects x_n's sequentially, where n is the dummy index to mark the n^{th} iteration of sequential design.

2.3.1 Gaussian Process

For the response modeling aspect, GP regression, or kriging, has emerged as the most popular nonparametric approach for both deterministic and stochastic black-box functions thanks to their flexibility, analytical tractability and superior empirical performance. The kriging methodology has been very popular in various engineering disciplines for approximating the response of deterministic computer experiments, see Santner et al. [93]. In the reliability analysis context where the performance function can be evaluated via deterministic experiments, the estimation of the safe zone (more precisely its volume $\mu(S)$) was carried out in [6, 74, 30, 83, 5, 35]. All of these works employed a Gaussian Process surrogate to model the performance function.

In recent years, kriging related research for stochastic simulation has flourished, leading to a large amount of theoretical and empirical works on a wide range of topics. A majority of these studies have been dedicated to efficiently metamodeling the mean response surface implied by a

stochastic simulation [103, 1, 76]. There also exist research studies that focus on either jointly metamodeling the underlying mean and simulation variance of the response surfaces [13, 92, 9] or approximating quantile-based response surfaces [110, 21, 82]. Both of them aim at better coping with heteroscedasticity present in the stochastic simulation observations. As an extension, kriging is also widely used in bandit optimization [51, 97, 57, 53] and level set estimation [17, 38, 44, 83, 88]. Bayesian Optimization (BO) refers to a suite of techniques for bandit optimization where GP surrogates are used to model the underlying function f.

2.3.2 One-stage Design of Experiments

To build a high-quality metamodel such as kriging with a given simulation budget, a carefully designed simulation experiment is required. Literature on experimental designs for deterministic computer experiments proposed various design schemes, including Latin hypercube designs (LHD) [70], orthogonal array based LHD [79, 101], uniform designs [33], space-filling designs [50], and maximum entropy designs [95]. See [56] for details. All of these experimental designs choose all inputs at once, and are named as *One-stage* DoE frameworks. The One-stage DoE frameworks are efficient to carry out in offline simulations when the total budget N is known. However, in online simulations where the value of N is not given in advance, it is infeasible to apply the One-stage DoE frameworks. Besides, they choose all inputs up front, and do not make use of any information from the estimate. Therefore, these methodologies may waste simulations in less "informative" (or more certain) regions, like regions far away from the zero-contour in level set estimation or the optima in bandit optimization, where the sign of f is certainly to be positive or negative, or the value of f is much less than the minimum. In this thesis, One-stage DoE strategies are implemented to choose the initial input set, which is a small fraction of the total budget N, and provide an initial estimate of the level sets.

2.3.3 Sequential Design

Compared to the aforementioned non-sequential designs that choose all inputs up front, sequential designs offer a huge advantage in that they improve budget allocation efficiency by learning information from previous simulations and allocating the remaining simulation budget more wisely in the most uncertain region. GPs are well suited for sequential design by offering a rich uncertainty quantification aspect that can be (analytically) exploited to construct information-theoretic DoE heuristics. In a nutshell, in sequential design, at step n the GP paradigm constructs a metamodel

$$f^{(n)} = f|\mathbf{x}_{1:n}, \mathbf{y}_{1:n},$$

with $\mathbf{x}_{1:n}$ as the inputs chosen after n^{th} iteration in sequential design and $\mathbf{y}_{1:n}$ as their observations. Since at each iteration we choose one input, and at each input we simulate one observation, there are n inputs for $\mathbf{x}_{1:n}$ and n observations $\mathbf{y}_{1:n}$. The metamodel is then used to guide the selection of x_{n+1} and also to construct the estimate $\hat{S}^{(n)}$. The standard framework is to develop an acquisition function $\mathcal{I}_n(x)$ that quantifies the value of information from taking a new sample at input x conditional on an existing dataset $(x_{1:n}, y_{1:n})$ and then to myopically maximize $\mathcal{I}_n$:

$$x_{n+1} = \arg\max_{x \in D} \mathcal{I}_n(x). \tag{2.7}$$

GP metamodels have been shown to especially shine, not least because they organically match the sequential adaptive designs typically utilized. In this section, we provide a brief overview of the main categories of the acquisition function $\mathcal{I}_n(x)$ in existing literature. The first two algorithms, Expected Improvement (EI) and Gaussian Process Upper Confidence Bound (GP-UCB), were originally designed for Bayesian Optimization. Although BO is not the main focus of this thesis, both strategies were extended to level set estimation in later works. The other

four algorithms, Entropy, targeted Integrated Mean Square Error (tIMSE), Stepwise Uncertainty Reduction (SUR) and *Vorob'ev* deviation were originally proposed for level set estimation.

Expected Improvement. The respective Expected Improvement schemes form a major feature of the GP ecosystem. Expected Improvement was firstly proposed by Jones et al. [51] to solve efficient global optimization of expensive black-box functions. The idea of exploitation (sampling where the underlying function f is minimized) and exploration (sampling where prediction error is high) was first investigated in sequential design. They argued that the key to using metamodeling surfaces for global optimization lies in balancing the need to exploit the approximating surface with the need to improve the approximation. Huang et al. [45] extended the EI criterion to noisy observations.

Although the EI criterion was firstly proposed for Bayesian Optimization, it was later extended to solve level set estimation. The expected improvement criterion in [51] provides an indication of how much the true value of the response at an input can be expected to be less than the current best solution. It therefore makes little sense to apply this to the level set estimation problem where the goal is to estimate the level set or the zero-contour. Bichon et al. [7] proposed Expected Feasibility Function (EFF), which provides an indication of how well the true value of the response is expected to be on the target zero-contour. Similarly, Ranjan et al. [89] proposed an expected improvement function for level set estimation, which chooses the input that is the closest to the zero-contour.

Gaussian Process Upper Confidence Bound. Srinivas et al. [97] resolved the open problem of deriving regret (difference between true and estimated maximum) bounds for Bayesian Optimization, and analyzed an intuitive Gaussian Process Upper Confidence Bound algorithm. Their algorithm established a novel connection between GP surrogates and sequential design, and chooses the inputs whose upper confidence bound of the response f is the largest.

The same as EI criterion, GP-UCB was also designed for Bayesian Optimization. It is not reasonable to measure the upper confidence bound of the response f itself. Instead, for level

set estimation, we use an approximate lower confidence bound of the absolute value of f as the measurement. The smaller the lower confidence bound is, the closer the input is to the zero-contour. In fact, Bryan et al. [16] developed similar criterion even earlier, *straddle*, for deterministic level set estimation. Gotovos et al. [38] and Wang et al. [107] further provided theoretical guarantees about its performance in stochastic setting.

Entropy. Bryan et al. [16] described several entropy-related strategies for level set estimation in deterministic setting. They proposed to choose the input with the largest probability of being misclassified to the wrong level (i.e., $x \in S$ while $\hat{f}^{(n)} < 0$ or $x \in D$ while $\hat{f}^{(n)} > 0$), the input with the largest entropy, the input with the largest information gain, or the input with the largest variance. The first three strategies select the same input since the second and third measurements are monotone functions of the first measurement. Similar strategies to estimate the probability of failure were also investigated by Ranjan et al. [89], Bichon et al. [7] and Echard et al. [31]. The basic idea is to select the input x having the greatest misclassification probability. There are also similar strategies for Bayesian Optimization, say, Entropy Search (ES), proposed by Hennig and Schuler [42]. ES selects inputs to produce greedy maximization of the mutual information between the optima x_* and the observation y. It is argued that the first four strategies either solely focus on exploration of the space, or exploitation of the targeted zero-contour. Purely exploration-based strategies lead to over-exploration of the edges of the input space D and lack of exploration of the zero-contour or regions of interest [16]. One problem for the purely exploitation-based strategies is that there are infinite solutions to optimization of the acquisition function when the zero-contour is not trivial. It is also suggested to use the combination (say, product or weighted sum) of any two aforementioned entropy-related measures together as the acquisition function.

Note that *entropy* or misclassification probability is originally used for classification problem. However, the goal of level set estimation is to learn where the response is positive, which is equivalent to classifying each input x either to S or to N. Instead of the numerical value of the

response itself, we can transfer the problem into learning the sign of the response and thus the level set estimation problem benefits from classification-related methods. This also motivates our work in Section 3.3.

Targeted Integrated Mean Square Error. The other criterion, targeted IMSE, proposed by Picheny et al. [83], stemmed from the IMSE criterion for surface metamodeling. The main idea of this work is that the surrogate does not need to be globally accurate, but only in some region of interest. This is implemented by adding weight to the IMSE: region close to the targeted zero-contour gets more weight, while other regions get less. The tIMSE criterion operates a trade-off between global uncertainty reduction (high variance) and exploration of target regions (close to the zero-contour) via adding weights to regions of interest.

Stepwise Uncertainty Reduction. Stepwise Uncertainty Reduction strategy was firstly used in the field of Bayesian Optimization: Villemonteix et al. [106] proposed the Informational Approach to Global Optimization (IAGO), where the Shannon entropy of the minimizer is used. Frazier et al. [34] proposed *knowledge gradient policies* (KG), which uses utility rather than cost as the acquisition function. Specifically, the KG strategy aims to myopically maximize the global learning rate about the optima x_*. Similarly, Probability of Improvement (PI) over the current best observation [62] maximizes the posterior probability of value of underlying function f at the next input being greater than the current maximum observation. Criteria more targeted to reduce the uncertainty about the level set S itself were first developed by Bect et al. [6] using the concept of SUR. Their objective is to obtain an approximation of positive level set S and measure the uncertainty of this approximation. Besides the local probability of being misclassified into the wrong level, SUR strategies aim at minimizing the global misclassification probability or level set uncertainty. Different from the entropy-based strategies that focus on the *marginal distribution* at the candidate input $x \in D$, both tIMSE and SUR strategies require integration of the local measures. Consequently, they are more expensive to compute than the entropy-based strategies.

***Vorob'ev* Deviation.** All of the strategies mentioned so far for level set estimation are designed for deterministic settings. Recently, more studies focus on stochastic settings. One category of the literature for stochastic experiments still focuses on developing the acquisition functions for sequential design, and obtain similar expressions mentioned, as GP-UCB related strategies for noisy level set estimation in Gotovas et al. [38]. Others focus on developing robust metamodels/criteria/inference to handle the heteroscedasticity, see [82, 8]. Further criteria using tools from random set theory were developed [24, 2] for noisy observations. Specifically, those works use the notions of *Vorob'ev* expectation and *Vorob'ev* deviation to choose inputs that minimize the posterior expected distance in measure between the level set S and its estimate $\hat{S}$. *Vorob'ev* expectation and *Vorob'ev* deviation were firstly proposed to estimate and quantify the uncertainty of the level set [24]. This is the first approach that focused on the set itself rather than solely on its volume. It was proved that when the function is actually a GP realization, the *Vorob'ev* deviation converges to zero in infill asymptotics. The *Vorob'ev* deviation was then used as an acquisition function to guide selection of inputs in sequential design for noisy level set estimation [2]. It is equivalent to choose the input which produces the smallest distance between the true level set S and its estimate $\hat{S}$ after adding this input. This leads to the same expression with the acquisition function of SUR, which is basically used for deterministic experiments. This approach is computationally expensive however, and requires conditional simulations of the posterior Gaussian field. Azzimonti et al. [2] also proposed the marginal form, which is cheaper to compute.

2.3.4 Efficient Sequential Designs

Aforementioned strategies with Gaussian Process have been used widely for surface metamodeling, Bayesian Optimization and level set estimation. However, most of them are cumbersome in calculation due to high computational complexity, expensive function evaluation,

and the demand for numerical integration. Besides, they choose only one input at each step, while with development of modern computing machines, we can afford fast computation in parallel. Accordingly, Chevalier et al. [23] proposed two multi-sampling SUR criteria based on the original SUR strategies, where multiple inputs are selected at each iteration.

In Bayesian Optimization, how to strike a balance between the lower cost and inaccurate evaluation of f versus the higher cost and accurate evaluation of f, namely, *multi-fidelity* optimization, has recently gained attention [53, 54, 55, 60, 28]. As suggested by the name, these methods assume we have access to lower fidelity approximations to f which can be evaluated instead of f. The lower the fidelity, the cheaper the evaluation and the less accurate the observation. The timing cost for expensive function evaluation has been saved in sacrifice of accuracy by choosing inputs from a lower fidelity which is cheaper to access but produces a less accurate observation. For example, when optimizing the set of hyper-parameters for a complex machine learning system, its loss (response in this example) can be approximated using a smaller validation set. Consequently, the evaluation is cheaper but less accurate. The fidelity refers to the size of the validation set in this example. Kandasamy et al. [55] extended the multi-fidelity optimization to continuous fidelity settings.

Batching/replication is another trick which benefits both prediction accuracy and computational efficiency perspectives in noisy computer experiments. It is known in the stochastic simulation community as nested Monte Carlo. Re-using the same input to generate multiple outputs allows for a Law of Large Numbers (LLN) averaging which can be analytically combined with the GP predictive equations to keep the computational complexity as a function of k (number of unique inputs) rather than of the capital-N (number of simulator calls). The seminal technique of stochastic kriging [1] shows that these computational savings are exact assuming the GP hyperparameters, in particular the noise variance $\tau^2(x)$, are known. Such batching becomes critical in the use of GP models in our motivating application of solving optimal stopping problems via Regression Monte Carlo, where tens of thousands of simulations

are called for.

In the classical setup, the metamodeling objective is to learn the mean response over the entire domain [58, 59, 22], whereby, modulo heteroscedastic noise, one expects to utilize the same batching level across all inputs, i.e. splitting the total budget

$$N = k \times r$$

into k batches of r replicates at inputs $\bar{x}_1, \ldots, \bar{x}_k$. It is especially beneficial to investigate allocation of replicates over inputs when the noise is heteroscedastic over the entire space: basically regions with large uncertainty need more simulations to smooth the noise [10]. See Ankenman et al. [1] for a discussion of how to pick k for a given budget N, as well as some proposals for handling non-constant $\tau^2(x)$. It is demonstrated that how replicates could improve the accuracy of estimating surrogate surface in noisy computer experiments via stochastic kriging. Replication in this paper is mainly used to estimate the empirical variance for noise. Further efforts work on augmenting the replication in a few (usually two or three) stages [61, 86, 72] for Bayesian Optimization or surface metamodeling.

Allocation of replicates over inputs is also investigated in stochastic optimization, known as *Optimal Computing Budget Allocation* (OCBA). The goal of OCBA is to obtain the highest decision quality using a fixed simulation budget, and the basic idea is that regions of interest, for example, regions close to the optima in optimization, should receive more simulations. Rules of sequential allocation over a fixed design set were proposed by Chen et al. [20]. The main difference between OCBA and the work in this thesis is that we assimilate the design set via sequential design while in the OCBA literature, the designs are fixed initially, and only more simulations are allocated on the designs sequentially by iterations; Another difference is that for OCBA, no surrogate is used to estimate the underlying function f. Instead the distribution of f is approximated via Monte Carlo mean and variance through Large Law of Numbers. Similar

adaptive reallocation framework was also investigated in [66] to estimate the portfolio tail risk.

Recent literature combines design of experiments with replication together, solving both where to select the next input and how many replicates should be performed at each input together in sequential design framework. Since the sequential design frameworks imply preferentially sampling a small portion of the input space for level set estimation or BO—the neighborhood of the maximum, or the neighborhood of the desired contour—the exploration-exploitation paradigm leads to increasingly concentrated designs. Such concentration suggests to adaptively determine the amount of batching. Intuitively, replication should be low for more exploratory sites and should rise in the neighborhood of interest, where we replicate to achieve computational savings. Indeed, the intrinsic cost of replication is linked to the variability of the response at the respective inputs, which will be minimal if the inputs are very close together. From a different perspective, replication trades off costly, precise outputs (large batch size) vis-a-vis cheap outputs with low signal-to-noise ratio (small batch size).

One solution to combine DoE with replication is to firstly select the next input, and then determine how many new simulations should be allocate to this input. Picheny et al. [82] presented a strategy to optimize noisy computer experiments: select a new input via a quantile-based criterion (Expected Quantile Improvement) and then keep allocating extra simulations to it until the stopping criterion is satisfied. Jalali et al [47] provided a comparison between several related algorithms with replication at inputs for simulation optimization with heterogeneous noise. Binois et al [9, 10] developed a lookahead-based sequential design scheme for surface metamodeling that determines if a new simulation should be at an existing input or a new input. Chen and Zhou [22] proposed another solution, *Balanced Stepwise Approximate Optimal Design* (BSAO) strategy, for surface metamodeling, which adaptively balances exploration (allocate extra simulations to existing inputs) and exploitation (allocate extra simulations to one new input) with a given number of new simulations at each step. Both replication related and parallel computing related strategies reduce the computational complexity of the problem

from the number of total simulations to the number of iterations/unique inputs. For parallel strategy proposed in [23], new simulations at each step are allocated evenly over the new inputs: one input get one simulation, and the design size augments by the number of new simulations allocated at each step; on the other hand, for replication strategy proposed in [22], the design size either remains the same or augments by 1 at each step.

2.4 Summary of Contributions

As discussed in previous sections, challenges lie in two aspects when we reconstruct the level set S: the first challenge is how to construct a robust surrogate to model the response given noisy observations. Most of the cited papers consider only the deterministic setting without any simulation noise. While there are plenty works discussing how to handle the simple versions (with constant or prespecified Gaussian noise), the literature on GP surrogates for complex stochastic simulators remains incomplete. Clear analysis comparing all these choices in the stochastic setting is currently lacking. Another challenge is the computational efficiency for stochastic computer simulations, especially when the simulation size N is large. For GP metamodels to be fast, it is imperative to keep the respective simulation size N manageable. In particular, unless the simulator is truly expensive or the input domain is vast, the typical recommendation is to restrict to hundreds of inputs, $N \ll 10^3$ [9]. This creates a major tension as frequently the stochastic simulator has low signal-to-noise ratio or a complex noise structure. A prototypical example that comes from our motivation benchmarks is where the simulator $Y(x) = F(X_{[0,\Delta t]})|_{X_0=x}$ involves functionals of a continuous-time Markov chain or stochastic differential equation solution (X_t), whereby the stochasticity tends to dominate the trend/drift term for short Δt, and moreover simulation noise is non-Gaussian and state-dependent (heteroscedastic).

With respect to the first challenge, the first part of this research focuses on reconstructing

S via estimating the latent function f with different metamodels in stochastic settings. As discussed in Section 2.3, the contributions are divided into two parts corresponding to the two steps to reconstruct S: to present a comprehensive assessment of GP-based metamodels for stochastic contour-finding, and to propose and develop sequential design strategies for the GP-based metamodels.

In the first respect, our analysis complements the work of Picheny et al. [84] and Jalali et al. [48], who benchmarked GP metamodels for Bayesian optimization. Several works focused on heteroscedastic simulation variance; see the Stochastic Kriging approach of Ankenman et al. [1] and the earlier works by two of the authors [98, 75]. In this research, we instead target the non-Gaussian aspects of observations for level set estimation, in particular the likely heavy-tailed property. This issue is fundamental to any realistic stochastic simulator where there is no justification for assuming Gaussian-distributed $\epsilon(x)$ (as opposed to the physical experimental setup where ϵ represents observation noise and is expected to be Gaussian thanks to the central limit theorem). This motivates us to study *alternative GP-based metamodels* for learning $\hat{S}$ that are more robust to non-Gaussian ϵ in (2.1). In parallel, we investigate which of the contour-finding heuristics outlined above perform best in such setups.

To stay within the overarching sequential design paradigm, we continue to work with a GP-based setup but investigate several modifications that are relevant for learning $\hat{S}$.

- To relax the Gaussian noise assumption, we investigate t-observation GPs [109, 52]; use of the Student-t likelihood nests both the heavy-tailed and Gaussian cases.

- As another non-Gaussian specification we consider Student-t processes (TPs) [94, 107] that are also resistant to observation outliers.

- To target the classification-like underlying objective (2.2), we consider the use of classification GPs that model the sign of the response $Y(x)$ via a probit logistic model driven by a latent GP $Z(\cdot)$: $\mathbb{P}(Y(x) > 0|x) = \text{probit}(Z(x))$. Deployment of the logistic regression

is expected to "wash out" non-Gaussian features in $\epsilon(x)$ beyond its effect on the sign of the observations.

- In a different vein, to exploit a structure commonly encountered in applications where the level set S is *connected*, we study the performance of *monotone* GP regression/classification metamodels [91] that force f (or Z) to be monotone in the specified coordinates.

This research is mainly driven by the effect of noise on level set estimation. The combination of sequential designs with the GP frameworks targets on quantifying the uncertainty and constructing a robust surrogate. In the context of GPs, this means accurate inference of the response and sampling noise that in turn drive the posterior mean $\hat{f}(x)$ and the posterior GP variance $s(x)^2$. Both of the latter ingredients are needed to blend the exploitation objective to locally learn the contour ∂S and to explore less-sampled regions. These issues drive our choices of the metamodels and also factor in developing the respective acquisition functions $\mathcal{I}_n(x)$; see cf. Section 4.1. On the latter front we consider four choices (MCU, cSUR, tMSE, ICU), including heuristics that depend only on the posterior standard deviation $s^{(n)}(\cdot)$, as well as those that anticipate information gain from sampling at x_{n+1} via the look-ahead standard deviation $s^{(n+1)}(\cdot)$. Because in the non-Gaussian GPs $s^{(n+1)}$ depends on $Y(x_{n+1})$, we develop tractable approximations $\hat{s}^{(n+1)}$ for that purpose, see Propositions 4.2.2-4.2.3-4.2.5.

The other challenge for noisy level set estimation is the concern about computational efficiency as the simulation size N explodes in stochastic computer experiments. This motivates *adaptively batched* designs, where r is input-dependent. While this idea was investigated for surface metamodeling with IMSE minimization [1, 10], neither of these fully reveal the underlying tension between exploration (replicate less, larger metamodel overhead) and exploitation (replicate more, generate computational savings). In this research we propose several schemes that explicitly focus on this issue. To evaluate them we concentrate on the problem of level set estimation where the zero-contour is adaptively learned through the sequential design but retains

a spatial structure (unlike Bayesian Optimization where convergence to the single input yielding the global maximum is desired). Consequently, we expect a complex interaction between the selection of inputs and the respective replication amounts. In this context, our main contribution is to extend the paradigm of Expected Improvement to include sequential selection of both the input locations x_n and the replication counts r_n. We benchmark the proposed algorithms and show that they provide significant savings compared to the naive fixed-batching approach. In particular, we are able to obtain schemes that reduce $N \simeq 10^5$ simulations to efficient replicated designs of just a few hundred unique inputs.

Beyond benchmarking the developed algorithms on several synthetic examples, we also implement them for the motivating application of valuation of Bermudan options. In the latter context, the Regression Monte Carlo paradigm is used to provide a simulation-based algorithm that hinges on recursive estimation of certain level sets that correspond to the stopping boundaries. Building upon the successful use of GP surrogates for RMC [64, 67], we demonstrate that adaptive batching significantly speeds up this approach and makes it even more efficient and scalable.

To recap, the main contributions of this thesis can be traced along six directions. First, we investigate two ways to handle heavy-tailed simulation noise via GP with t-observations and via TP. As far as we are aware, this is the first application of either tool in sequential design and level set estimation contexts. Second, we present an original use of monotonic GP metamodels for level set estimation. This idea is related to a gray-box approach that aims to exploit known structural properties of f (or S) so as to improve on the agnostic black-box strategies. Third, we analyze the performance of classification GP metamodels for contour-finding. This context offers an interesting and novel comparison between regression and classification approaches benchmarked against a shared loss function. Fourth, we develop and implement approximate *look-ahead* formulas for all our metamodels that are used for the evaluation of acquisition functions. To our knowledge, this is the first presentation of such

formulas for non-Gaussian GPs, as well as TPs. Fifth, beyond the metamodels themselves, we also provide a detailed comparison among the proposed acquisition functions, identifying the best-performing combinations of $\mathcal{I}(\cdot)$ and metamodel $\hat{f}$ and documenting the complex interplay between design geometry and surrogate architecture. Sixth, we propose four adaptive batching algorithms for two GP metamodels, which determine both the inputs x_n and its batch size r_n, and reduce the complexity of computation while maintaining the metamodeling fidelity.

Chapter 3

Level Set Estimation with Gaussian Process Surrogates

3.1 Gaussian Process Regression with Gaussian Noise

We begin by discussing regression frameworks for level set estimation that target learning the latent $f(\cdot)$. The Gaussian process paradigm treats f as a random function whose posterior distribution is determined from its prior and the collected samples $\mathcal{A}_n \equiv \{(x_i, y_i), 1 \leq i \leq n\}$. A priori, we view $f(\cdot) \sim GP(m(\cdot), K(\cdot, \cdot))$ as a realization of a Gaussian process completely specified by its mean function $m(x) := \mathbb{E}[f(x)]$ and covariance function $K(x, x') := \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]$.

In the classical case [109], the noise distribution is homoskedastic Gaussian $\epsilon(x) \sim \mathcal{N}(0, \tau^2)$, and the prior mean is zero, $m(x) = 0$. Given observations $\mathbf{y}_{1:n} = [y_1, ..., y_n]^T$ at inputs $\mathbf{x}_{1:n} = [x_1, \ldots, x_n]^T$, the conditional distribution $f|\mathcal{A}_n$ is then another Gaussian process, with posterior marginal mean $\hat{f}^{(n)}_{\text{Gsn}}(x_*)$ and covariance $v^{(n)}_{\text{Gsn}}(x_*, x'_*)$ given by (throughout we use

subscripts to indicate the metamodel type, e.g., Gsn for Gaussian noise)

$$\hat{f}^{(n)}_{\text{Gsn}}(x_*) = k(x_*)[\mathbf{K} + \tau^2\mathbf{I}]^{-1}\mathbf{y}_{1:n}, \tag{3.1}$$

$$v^{(n)}_{\text{Gsn}}(x_*, x'_*) = K(x_*, x'_*) - k(x_*)[\mathbf{K} + \tau^2\mathbf{I}]^{-1}k(x'_*)^T, \tag{3.2}$$

with the $1 \times n$ vector $k(x_*)$ and $n \times n$ matrix $\mathbf{K}$ defined by

$$k(x_*) := K(x_*, \mathbf{x}_{1:n}) = [K(x_*, x_1), ..., K(x_*, x_n)], \quad \text{and} \quad \mathbf{K}_{i,j} := K(x_i, x_j). \tag{3.3}$$

The posterior mean $\hat{f}^{(n)}_{\text{Gsn}}(x_*)$ is treated as a point estimate of $f(x_*)$ and the posterior standard deviation $s^{(n)}_{\text{Gsn}}(x_*)^2 = v^{(n)}_{\text{Gsn}}(x_*, x_*)$ as the uncertainty of this estimate. We use $\mathbf{f}$ to denote the random posterior vector $f(\mathbf{x}_{1:n})|\mathcal{A}_n$.

Model Fitting: In our research, we model the covariance between the values of f at two inputs x and x' with the squared exponential (SE) function:

$$K_{\text{se}}(x, x') := \sigma^2_{\text{se}} \exp(-2l^2), \tag{3.4}$$

with $l = \sqrt{\sum_{i=1}^{d} \frac{(x^i - x'^i)^2}{\theta_i^2}}$, defined in terms of the hyperparameters $\boldsymbol{\vartheta} = \{\sigma_{\text{se}}, \theta_1, ..., \theta_d, \tau\}$ known as the process variance and length-scales, respectively. Simulation variance τ is also treated as unknown and part of $\boldsymbol{\vartheta}$. The length-scales $\{\theta_1, ..., \theta_d\}$ control the smoothness of the GP. When θ is large, the samples drawn from the GP tend to be smoother, while otherwise, they are wiggling. Other choices include Matérn $3/2$ and $5/2$ kernels:

$$K_{3/2}(x, x') := \sigma^2_{\text{se}}(1 + \sqrt{3}l) \exp(-\sqrt{3}l), \tag{3.5}$$

$$K_{5/2}(x, x') := \sigma^2_{\text{se}}(1 + \sqrt{5}l + \frac{5}{3}l^2) \exp(-\sqrt{5}l), \tag{3.6}$$

giving once and twice differentiable functions, respectively. Several common ways exist for

estimating hyperparameters $\boldsymbol{\vartheta}$ in covariance function. Within a Bayesian approach we integrate against the prior $p(\boldsymbol{\vartheta})$ using

$$p(\mathbf{f}|\mathbf{y}_{1:n}, \mathbf{x}_{1:n}, \boldsymbol{\vartheta}) = \frac{p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \mathbf{f})p(\mathbf{f}|\boldsymbol{\vartheta})}{p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \boldsymbol{\vartheta})}, \tag{3.7}$$

where $p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \mathbf{f})$ is the likelihood and $p(\mathbf{f}|\boldsymbol{\vartheta})$ is the latent function prior. Notice that following the Gaussian noise assumption, the likelihood $p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \mathbf{f})$ is Gaussian. With a Gaussian prior $p(\mathbf{f}|\boldsymbol{\vartheta})$, the posterior $p(\mathbf{f}|\mathbf{y}_{1:n}, \mathbf{x}_{1:n}, \boldsymbol{\vartheta})$ is tractable and also follows a Gaussian distribution. The normalizing constant in the denominator $p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \boldsymbol{\vartheta})$ is independent of the latent function and is called the marginal likelihood, given by

$$p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \boldsymbol{\vartheta}) = \int p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \mathbf{f})p(\mathbf{f}|\boldsymbol{\vartheta})d\mathbf{f}. \tag{3.8}$$

One may similarly express the posterior over the hyperparameters $\boldsymbol{\vartheta}$, where $p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \boldsymbol{\vartheta})$ plays the role of the likelihood. To avoid expensive MCMC integration, we use the maximum a posteriori probability (MAP) estimate $\hat{\boldsymbol{\vartheta}}$ which maximizes the likelihood over $\boldsymbol{\vartheta}$:

$$\hat{\boldsymbol{\vartheta}} = \arg\max_{\boldsymbol{\vartheta}} p(\boldsymbol{\vartheta}|\mathbf{y}_{1:n}, \mathbf{x}_{1:n}) = \arg\max_{\boldsymbol{\vartheta}} \frac{p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \boldsymbol{\vartheta})p(\boldsymbol{\vartheta})}{p(\mathbf{y}_{1:n}|\mathbf{x}_{1:n})}. \tag{3.9}$$

Given the estimated hyperparameters $\hat{\boldsymbol{\vartheta}}$, we take the posterior of f as $p(\mathbf{f}|\mathbf{y}_{1:n}, \mathbf{x}_{1:n}, \hat{\boldsymbol{\vartheta}})$.

3.2 Gaussian Process Regression with Student t-Noise

Taking the noise term $\epsilon(x)$ as Gaussian is widely used since the marginal likelihood is then analytically tractable. In a stochastic simulation setting however, the exact distribution of the outputs relative to their mean is unknown and often is clearly non-Gaussian. A more robust choice is to assume that $\epsilon(x)$ has a Student-t distribution [52]. In particular, this may work

better when the noise is heavy-tailed by making inference more resistant to outliers [77]. In the resulting t-GP formulation $\epsilon(x)$ is assumed to be t-distributed with variance τ^2 and $\nu > 2$ degrees of freedom (the latter is treated as another hyperparameter). The marginal likelihood of observing $\mathbf{y}_{1:n}$ can be written as

$$p_{t\mathrm{GP}}(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \mathbf{f}) = \prod_{i=1}^{n} \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)\sqrt{\nu\pi}\sigma_n} \left(1 + \frac{(y_i - f_i)^2}{\nu\sigma_n^2}\right)^{-(\nu+1)/2}, \tag{3.10}$$

where $\Gamma(\cdot)$ is the incomplete Gamma function. The likelihood $p_{t\mathrm{GP}}(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}, \mathbf{f})$ in (3.7) is no longer Gaussian, and integrating (3.10) against the Gaussian prior $p(f|\boldsymbol{\vartheta})$ is intractable; we therefore use the Laplace approximation (LP) method [108] to calculate the posterior. A second-order Taylor expansion of $\log p_{t\mathrm{GP}}(\mathbf{f}|\mathbf{x}_{1:n}, \mathbf{y}_{1:n})$ around its mode, $\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n)} := \arg\max_{\mathbf{f}} p_{t\mathrm{GP}}(\mathbf{f}|\mathbf{x}_{1:n}, \mathbf{y}_{1:n})$, gives a Gaussian approximation

$$p_{t\mathrm{GP}}(\mathbf{f}|\mathbf{x}_{1:n}, \mathbf{y}_{1:n}) \approx q_{t\mathrm{GP}}(\mathbf{f}|\mathbf{x}_{1:n}, \mathbf{y}_{1:n}) = \mathcal{N}\left(\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n)}, \boldsymbol{\Sigma}_{t\mathrm{GP}}^{-1}\right), \tag{3.11}$$

where $\boldsymbol{\Sigma}_{t\mathrm{GP}}^{-1}$ is the Hessian of the negative conditional log posterior density at $\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n)}$:

$$\boldsymbol{\Sigma}_{t\mathrm{GP}} = -\nabla^2 \log p_{t\mathrm{GP}}(\mathbf{f}|\mathbf{x}_{1:n}, \mathbf{y}_{1:n})|_{\mathbf{f}=\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n)}} = \mathbf{K}^{-1} + \mathbf{W}_{t\mathrm{GP}}, \tag{3.12}$$

and $\mathbf{W}_{t\mathrm{GP}} = -\nabla^2 \log p_{t\mathrm{GP}}(\mathbf{y}_{1:n}|\mathbf{f}, \mathbf{x}_{1:n})|_{\mathbf{f}=\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n)}}$ is diagonal, since the likelihood factorizes over observations.

Using (3.11), the approximate posterior distribution $f(x_*)|\mathcal{A}_n \sim \mathcal{N}(\hat{f}_{t\mathrm{GP}}^{(n)}(x_*), s_{t\mathrm{GP}}^2(x_*))$ is also Gaussian defined by its mean $\hat{f}_{t\mathrm{GP}}^{(n)}(x_*)$ and covariance $v_{t\mathrm{GP}}^{(n)}(x_*, x'_*)$:

$$\hat{f}_{t\mathrm{GP}}^{(n)}(x_*) = k(x_*)\mathbf{K}^{-1}\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n)}, \tag{3.13}$$

$$v_{t\mathrm{GP}}^{(n)}(x_*, x'_*) = K(x_*, x'_*) - k(x_*)(\mathbf{K} + \mathbf{W}_{t\mathrm{GP}}^{-1})^{-1}k(x'_*). \tag{3.14}$$

Note the similarity to (3.1)–(3.2): with Student-t likelihood the mode $\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n)}$ plays the role of $\mathbf{y}_{1:n}$ and $\boldsymbol{W}_{t\mathrm{GP}}^{-1}$ replaces the noise matrix $\tau^2\mathbf{I}$. Critically, the latter implies that the posterior variance is a function of both designs $\mathbf{x}_{1:n}$ and observations $\mathbf{y}_{1:n}$.

3.3 Gaussian Process Classification

Our target in (2.2) is to learn where the mean response is positive, which is equivalent to classifying each $x \in D$ as belonging either to S or to N. Assuming that $\epsilon(x)$ is symmetric, $\{x \in S\} = \{f(x) \geq 0\} = \{\mathbb{P}(Y(x) > 0) > 0.5\}$. This motivates us to consider the alternative of directly modeling the response sign (rather than overall magnitude) via a classification GP model (Cl-GP) [108, 109]. The idea is to model the probability of a positive observation $Y(x)$ by using a probit logistic regression: $\mathbb{P}(Y(x) > 0|x) = \Phi(Z(x))$, with $\Phi(\cdot)$ the standard normal cumulative density function (CDF). The latent classifier function is taken as the GP $Z \sim GP(0, K(\cdot,\cdot))$. After learning Z we then set $\hat{S} = \{x : \hat{Z}(x) > 0\}$.

To compute the posterior distribution of Z conditional on $\mathcal{A}_n$, we use the fact that for an observation (x_i, y_i) and conditional on $z_i = Z(x_i)$ the likelihood of $y_i > 0$ is $\Phi(z_i)1_{\{y_i \geq 0\}} + (1 - \Phi(z_i))1_{\{y_i < 0\}}$. To simplify notation we use $\check{Y}(x) = \operatorname{sign} Y(x) \in \{-1, 1\}$ to represent the signed responses driving Cl-GP, leading to $p_{\mathrm{Cl}}(\check{\boldsymbol{y}}_{1:n}|\mathbf{z}, \mathbf{x}_{1:n}) = \prod_{i=1}^{n} \Phi(\check{y}_i z_i)$. The posterior of the latent $\mathbf{z} = Z(x_{1:n})$ is therefore

$$p_{\mathrm{Cl}}(\mathbf{z}|\mathbf{x}_{1:n}, \check{\boldsymbol{y}}_{1:n}) = \frac{p(\mathbf{z}|\mathbf{x}_{1:n}) \prod_{i=1}^{n} \Phi(\check{y}_i z_i)}{p(\check{\boldsymbol{y}}_{1:n}|\mathbf{x}_{1:n})}. \tag{3.15}$$

Similar to the t-GP, we use a Laplace approximation for the non-Gaussian $p_{\mathrm{Cl}}(\mathbf{z}|\mathbf{x}_{1:n}, \check{\boldsymbol{y}}_{1:n})$ in Eq. (3.15),

$$p_{\mathrm{Cl}}(\mathbf{z}|\mathbf{x}_{1:n}, \check{\boldsymbol{y}}_{1:n}) \approx q_{\mathrm{Cl}}(\mathbf{z}|\mathbf{x}_{1:n}, \check{\boldsymbol{y}}_{1:n}) = \mathcal{N}(\tilde{\mathbf{z}}^{(n)}, \boldsymbol{\Sigma}_{\mathrm{Cl}}^{-1}), \tag{3.16}$$

where we again use the mode $\tilde{\mathbf{z}}^{(n)} := \arg\max_{\mathbf{z}} p(\mathbf{z}|\mathcal{A}_n)$ and $\mathbf{\Sigma}_{\text{Cl}}$ is the Hessian of the negative log posterior at $\tilde{\mathbf{z}}^{(n)}$:

$$\mathbf{\Sigma}_{\text{Cl}} = -\nabla^2 \log p_{\text{Cl}}(\mathbf{z}|\mathcal{A}_n)\big|_{\mathbf{z}=\tilde{\mathbf{z}}^{(n)}} = \mathbf{K}^{-1} + \mathbf{V}, \tag{3.17}$$

and $\mathbf{V} = -\nabla^2 \log p(\check{\boldsymbol{y}}_{1:n}|\mathbf{z})|_{\mathbf{z}=\tilde{\mathbf{z}}^{(n)}}$ is diagonal with elements

$$v_i = \mathbf{V}_{ii} = -\frac{\partial^2}{\partial z_i^2} \log p(\check{y}_i|z_i)\big|_{z_i=\tilde{z}_i^{(n)}} = \frac{\phi(\tilde{z}_i^{(n)})^2}{\Phi(\check{y}_i\tilde{z}_i^{(n)})^2} + \frac{\check{y}_i\tilde{z}_i^{(n)}\phi(\tilde{z}_i^{(n)})}{\Phi(\check{y}_i\tilde{z}_i^{(n)})}, \qquad i = 1, \ldots, n, \tag{3.18}$$

$\phi(\cdot)$ denoting the density of the standard normal distribution.

The posterior mean for $Z(\cdot)$ at x_* is expressed by using the GP predictive mean equation (5.1) and LP approximation (3.16):

$$\hat{z}^{(n)}(x_*) = k(x_*)\mathbf{K}^{-1}\tilde{\mathbf{z}}^{(n)}, \tag{3.19}$$

$$v_{\text{Cl}}^{(n)}(x_*, x'_*) = K(x_*, x'_*) - k(x_*)(\mathbf{K} + \mathbf{V}^{-1})^{-1}k(x'_*)^T. \tag{3.20}$$

We again see the same algebraic structure, with $\tilde{\mathbf{z}}^{(n)}$ a stand-in for $\mathbf{y}_{1:n}$ in (5.1) and $\mathbf{V}^{-1}$ a stand-in for $\tau^2\mathbf{I}$ in (5.2). Also note that we may formally link the Z of the Cl-GP metamodel to the GP observation Y used previously via the posterior probability that $x \in S$:

$$\begin{aligned}\mathbb{P}(Y(x) > 0|\mathcal{A}_n) &= \int_{\mathbb{R}} \Phi(z) p_{Z(x)}(z|\mathcal{A}_n) dz \\ &= \int \Phi(z)\phi\left(\frac{z - \hat{z}^{(n)}(x)}{s_{\text{Cl}}^{(n)}(x)}\right) dz = \Phi\left(\frac{\hat{z}^{(n)}(x)}{\sqrt{1 + s_{\text{Cl}}^{(n)}(x)^2}}\right).\end{aligned} \tag{3.21}$$

3.4 Gaussian Process with Monotonicity Constraint

Recall that since differentiation is a linear operator, the derivative of a GP f is another GP. Using $\mathbf{f}'$ as a shorthand notation for the gradient ∇f at locations $\mathbf{x}_{1:n}$, we have

$$\mathbb{E}[\partial_{x^j} f(x_*)|\mathcal{A}] = \frac{\partial \mathbb{E}[f(x_*)|\mathcal{A}]}{\partial x_*^j} = \frac{\partial \hat{f}(x_*)}{\partial x_*^j}; \tag{3.22}$$

$$\mathrm{Cov}(\partial_{x^j} f(x_*), f(x'_*)|\mathcal{A}) = K_{\mathbf{f}',\mathbf{f}}(x_*, x'_*) = \frac{\partial}{\partial x_*^j} K(x_*, x'_*) \quad \text{and} \tag{3.23}$$

$$\mathrm{Cov}(\partial_{x^j} f(x_*), \partial_{x^{j'}} f(x'_*)|\mathcal{A}) = K_{\mathbf{f}',\mathbf{f}'}(x_*, x'_*) = \frac{\partial^2}{\partial x_*^j \partial (x'_*)^{j'}} K(x_*, x'_*). \tag{3.24}$$

In addition to the data set $(\mathbf{x}_{1:n}, \mathbf{y}_{1:n})$, we now introduce virtual observations $(\mathbf{x}_v, \mathbf{y}_v)$ with the dummy responses $y_{v,i} \in \{-1, 1\} \times \{1, \ldots, d\}$ set according to whether f is required to be decreasing ($y_{v,i} = (-1, j)$) or increasing ($y_{v,i} = (+1, j)$) with respect to the jth input dimension at input $x_{v,i}$. The key "trick" is to use a probit likelihood $p(y_{v,i} = (+1, j)|\mathbf{x}_{1:n}, \mathbf{x}_v) = \Phi(\frac{1}{\eta}\partial_{x^j} f(x_{v,i}))$, where the small parameter η controls the strictness of the monotonicity constraint [91]. The probit function approaches the Heaviside step function when $\eta \to 0$ and forces the fitted $\partial_{x^j} \hat{f}(x_{v,i})$ (computed via (3.22)) to match during likelihood maximization the predetermined sign of $y_{v,i}$. An adaptive method to sequentially add the virtual inputs $\mathbf{x}_v$ is suggested in [91]. Note that if there are multiple monotonic dimensions, then the same $x_{v,i}$ might be reused multiple times to satisfy the constraints on $\partial_{x^j} \hat{f}$ across different j-coordinates, leading to a replicated design. We also remark that monotonic metamodels are more expensive to run, since they require the use of virtual observations that increase the effective sample size to $(\mathbf{x}_{1:n}, \mathbf{x}_v)$ and hence require inversion of larger $\boldsymbol{K}$-matrices.

The joint prior for $\mathbf{f}$ and its gradient $\mathbf{f}'$ is given by

$$p_{\mathrm{Mon}}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}' \end{bmatrix} \,\middle|\, \mathbf{x}_{1:n}, \mathbf{x}_v \right) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\text{joint}}), \tag{3.25}$$

where $\mathbf{K}_{\text{joint}} = \begin{bmatrix} K_{\mathbf{f},\mathbf{f}}(\mathbf{x}_{1:n}, \mathbf{x}_{1:n}) & K_{\mathbf{f},\mathbf{f}'}(\mathbf{x}_{1:n}, \mathbf{x}_v) \\ K_{\mathbf{f}',\mathbf{f}}(\mathbf{x}_v, \mathbf{x}_{1:n}) & K_{\mathbf{f}',\mathbf{f}'}(\mathbf{x}_v, \mathbf{x}_v) \end{bmatrix}$. Using Bayes rule, the joint posterior is then

$$p_{\text{Mon}}(\mathbf{f}, \mathbf{f}'|\mathbf{x}_{1:n}, \mathbf{y}_{1:n}, \mathbf{x}_v, \mathbf{y}_v) = \frac{p_{\text{Mon}}(\mathbf{f}, \mathbf{f}'|\mathbf{x}_{1:n}, \mathbf{x}_v) p(\mathbf{y}_{1:n}|\mathbf{f}) \prod_i \Phi\left(y_{v,i} \partial_{x_j} f(x_{v,i}) \frac{1}{\eta}\right)}{p(\mathbf{y}_{1:n}, \mathbf{y}_v|\mathbf{x}_{1:n}, \mathbf{x}_v)}. \quad (3.26)$$

Like for the classification GP to handle the non-Gaussian terms $p(y_{v,i}|\, \partial_{x_j} f(x_{v,i}))$ we approximate them with a local Gaussian likelihood

$$p(\mathbf{y}_v|\mathbf{f}') \approx q(\mathbf{y}_v|\mathbf{f}') = \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\text{Mon}}, \tilde{\boldsymbol{\Sigma}}_{\text{Mon}}). \quad (3.27)$$

We use the Expectation Propagation (EP) algorithm [73] to determine the vector of local means $\tilde{\mu}^i_{\text{Mon}}$, and the diagonal EP variance matrix $\tilde{\boldsymbol{\Sigma}}_{\text{Mon}}$, with local variances $(\tilde{\sigma}^i_{\text{Mon}})^2$. Details about the computation can be found in [91]. The approximate posterior to (3.26) is a product of Gaussian distributions and is simplified to

$$p_{\text{Mon}}(\mathbf{f}, \mathbf{f}'|\mathbf{x}_{1:n}, \mathbf{x}_v, \mathbf{y}_{1:n}, \mathbf{y}_v) \approx q_{\text{Mon}}(\mathbf{f}, \mathbf{f}'|\mathbf{x}_{1:n}, \mathbf{x}_v, \mathbf{y}_{1:n}, \mathbf{y}_v) = \mathcal{N}(\boldsymbol{\mu}_{\text{joint}}, \boldsymbol{\Sigma}_{\text{joint}}). \quad (3.28)$$

The covariance matrix is $\boldsymbol{\Sigma}^{-1}_{\text{joint}} = \mathbf{K}^{-1}_{\text{joint}} + \tilde{\boldsymbol{\Sigma}}^{-1}_{\text{joint}}$, with $\tilde{\boldsymbol{\Sigma}}_{\text{joint}} = \begin{bmatrix} \sigma^2\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \tilde{\boldsymbol{\Sigma}}_{\text{Mon}} \end{bmatrix}$, and the posterior mean is $\boldsymbol{\mu}_{\text{joint}} = \boldsymbol{\Sigma}_{\text{joint}} \tilde{\boldsymbol{\Sigma}}^{-1}_{\text{joint}} \tilde{\boldsymbol{\mu}}_{\text{joint}}$, with $\tilde{\boldsymbol{\mu}}_{\text{joint}} = \begin{bmatrix} \mathbf{y}_{1:n} \\ \tilde{\boldsymbol{\mu}}_{\text{Mon}} \end{bmatrix}$.

The posterior mean $\hat{f}_{\text{Mon}}(x_*)$ and posterior covariance $v_{\text{Mon}}(x_*, x'_*)$ for the M-GP metamodel

are

$$\hat{f}^{(n)}_{\text{Mon}}(x_*) = [k(x_*), K_{\mathbf{f},\mathbf{f}'}(x_*, \mathbf{x}_v)]\mathbf{K}^{-1}_{\text{joint}}\boldsymbol{\mu}_{\text{joint}}, \tag{3.29}$$

$$v^{(n)}_{\text{Mon}}(x_*, x'_*) = K_{\mathbf{f},\mathbf{f}}(x_*, x'_*) - [k(x_*), K_{\mathbf{f},\mathbf{f}'}(x_*, \mathbf{x}_v)](\mathbf{K}_{\text{joint}} + \tilde{\boldsymbol{\Sigma}}_{\text{joint}})^{-1} \begin{bmatrix} k(x'_*) \\ K_{\mathbf{f}',\mathbf{f}}(\mathbf{x}_v, x'_*) \end{bmatrix}, \tag{3.30}$$

analogously to the standard GP prediction equations (5.1) and (5.2).

In M-GP, replacing f with z and again applying the EP algorithm, we reach similar expressions for posterior mean/variance as in (3.29) and (3.30) and obtain the inference for monotonic classification GP (MCl-GP).

3.5 Student-t Process Regression with Student-t Noise

Instead of just adding Student-t likelihood to the observations as discussed in [109] and [52], Shah et al. [94] proposed t-processes (TPs) as an alternative to GPs, where they derived closed-form expressions for the marginal likelihood and posterior distribution of the t-process by imposing an inverse Wishart process prior over the covariance matrix of a GP model. They found the t-process to be more robust to model misspecification and to be particularly promising for BO. Moreover, Shah et al. [94] showed that TPs retain most of the appealing properties of GPs, including analytical expressions, with increased flexibility.

As noticed for example in [109], dealing with noisy observations is less straightforward with TPs, since the sum of two independent Student-t distributions has no closed form. Still, this drawback can be circumvented by incorporating the noise directly in the kernel. The corresponding data-generating mechanism is taken to be multivariate-t $\mathbf{y}_{1:n} \sim MVT\left(\nu, m(\mathbf{x}_{1:n}), \mathbf{K} + \tau^2\mathbf{I}\right)$, where the degrees of freedom are $\nu \in (2, \infty)$. The posterior predictive distribution is then

$f(x_*)|\mathcal{A}_n \sim \mathcal{T}\left(\nu+n, \hat{f}^{(n)}_{\mathrm{TP}}(x_*), v^{(n)}_{\mathrm{TP}}(x_*,x_*)\right)$, where [94]

$$\hat{f}^{(n)}_{\mathrm{TP}}(x_*) = k(x_*)[\mathbf{K}+\tau^2\mathbf{I}]^{-1}\mathbf{y}_{1:n}, \tag{3.31}$$

$$v^{(n)}_{\mathrm{TP}}(x_*,x'_*) = \frac{\nu+\beta^{(n)}-2}{\nu+n-2}[K(x_*,x'_*) - k(x_*)[\mathbf{K}+\tau^2\mathbf{I}]^{-1}k(x'_*)^T], \tag{3.32}$$

with

$$\beta^{(n)} := \mathbf{y}_{1:n}^\top \mathbf{K}^{-1}\mathbf{y}_{1:n}.$$

Comparing with the regular GPs, we have the same posterior mean $\hat{f}^{(n)}_{\mathrm{TP}}(x_*) = \hat{f}^{(n)}_{\mathrm{Gsn}}(x_*)$, but the posterior covariance now depends on observations $\mathbf{y}_{1:n}$ and is inflated by $v^{(n)}_{\mathrm{TP}}(x_*,x'_*) = \frac{\nu+\beta^{(n)}-2}{\nu+n-2}v^{(n)}_{\mathrm{Gsn}}(x_*,x'_*)$. Moreover, the latent function f and the noise are uncorrelated but not independent. As noticed in [94], assuming the same hyperparameters, as n goes to infinity, the above predictive distribution becomes Gaussian.

Inference of TPs can be performed similarly as for a GP, for instance based on the marginal likelihood:

$$p_{\mathrm{TP}}(\mathbf{y}_{1:n}|\mathbf{x}_{1:n},\boldsymbol{\vartheta}) = \frac{\Gamma(\frac{\nu+n}{2})}{((\nu-2)\pi)^{\frac{n}{2}}\Gamma(\frac{\nu}{2})}|\mathbf{K}|^{-1/2}\left(1+\frac{\mathbf{y}_{1:n}^\top\mathbf{K}^{-1}\mathbf{y}_{1:n}}{\nu-2}\right)^{-\frac{\nu+n}{2}}. \tag{3.33}$$

One issue is estimation of ν, which plays a central role in the TP predictions. We find that restricting ν to be small is important in order to avoid degenerating to the plain Gaussian GP setup.

3.6 Metamodel Performance for Level Set Inference

Sections 3.1, 3.2, 3.3, 3.4 and 3.5 introduce GP variants that are used to estimate the latent function f. After obtaining the estimated function $\hat{f}$ with any chosen metamodel, we then estimate the level set $\hat{S}$ through (2.4). To evaluate the performance of different metamodels, we

consider several metrics, which either measure the distance between the estimated level set $\hat{S}$ versus the true S, or the uncertainty of the estimate.

The first statistic is the error rate $\mathcal{ER}$, based on the loss defined in (2.3), comparing the resulting estimate $\hat{S}$ with the true S in terms of their symmetric difference. Let μ be a probability measure on the Borel σ-algebra $\boldsymbol{\mathcal{B}}(D)$ (e.g., $\mu = \text{Leb}_D$). The error rate $\mathcal{ER}$ measuring the distance between the level set S and its estimate $\hat{S}$:

$$\mathcal{ER} := \mu(S \Delta \hat{S}) = \int_{x \in D} \mathbb{I}\left[\text{sign}\, f(x) \neq \text{sign}\, \hat{f}(x)\right] \mu(dx). \tag{3.34}$$

For Cl-GP, we replace $f(x)$ with $z(x)$ in the above, namely, use $\mu(S \Delta \hat{S}) = \mu(\{x : \hat{z}(x) < 0 < z(x) \cup \hat{z}(x) > 0 > z(x)\})$. A related statistic is the bias $\mathcal{B}$, which is based on the *signed* (μ-weighted) difference between S and $\hat{S}$:

$$\mathcal{B} = \mu(S \backslash \hat{S}) - \mu(\hat{S} \backslash S) = \int_{x \in D} \left\{ \mathbb{I}[\hat{f}(x) < 0 < f(x)] - \mathbb{I}[\hat{f}(x) > 0 > f(x)] \right\} \mu(dx). \tag{3.35}$$

The error rate $\mathcal{ER}$ and bias $\mathcal{B}$ evaluate the accuracy of the point estimate $\hat{S}$ when the ground truth is known. In a realistic case study when the latter is unavailable, we replace $\mathcal{ER}$ by its empirical counterpart, based on quantifying the uncertainty in $\hat{S}$ through the associated uncertainty of $\hat{f}$. Following [2], we define the empirical error $\mathcal{E}$ as the expected distance in measure between the random set $S|\mathcal{A}$ and its estimate $\hat{S}$:

$$\mathcal{E} := \mathbb{E}\left[\mu(S \Delta \hat{S}) | \, \mathcal{A}\right] = \int_{x \in D} \bar{E}(x) \mu(dx), \tag{3.36}$$

with $\bar{E}(x)$ calculated by using (3.1) and (3.2):

$$\begin{aligned}\bar{E}(x) &:= \mathbb{E}\left[\mathbb{I}[\text{sign}\, f(x) \neq \text{sign}\, \hat{f}(x)]|\mathcal{A}\right] \\ &= \int_{\mathbb{R}} \mathbb{I}[\text{sign}\, f(x) \neq \text{sign}\, \hat{f}(x)] p(f(x)|\mathcal{A}) df(x) = \Phi\left(\frac{-|\hat{f}(x)|}{s(x)}\right). \end{aligned} \tag{3.37}$$

The local empirical error $\bar{E}(x)$ is the posterior probability of wrongly classifying x conditional on the training dataset $\mathcal{A}$. It is intrinsically tied to the point estimate $\hat{f}(x)$ and the associated posterior variance $s(x)^2$ through the Gaussian uncertainty quantification. For the TPs, the predictive distribution is Student-t, so that the Gaussian cdf Φ is replaced with the respective survival function.

Uncertainty Quantification: To quantify the overall uncertainty about S (rather than local uncertainty about $f(x)$), a natural criterion is the *volume* of the credible band $CI_{\partial S}$ that captures inputs x whose sign classification remains ambiguous given $\mathcal{A}$. A simple definition at a credibility level α (e.g., $\alpha = 0.05$) would be

$$CI_{\partial S}^{(n)} = \left\{x \in D : (\hat{f}^{(n)}(x) + z_{1-\frac{\alpha}{2}} s^{(n)}(x))(\hat{f}^{(n)}(x) - z_{1-\frac{\alpha}{2}} s^{(n)}(x)) < 0\right\}, \tag{3.38}$$

where $z_{1-\frac{\alpha}{2}}$ is the appropriate Gaussian/Student-t α-quantile. Thus (3.38) evaluates the region where the sign of f is nonconstant over the posterior α-CI of f. Heuristically however, $CI_{\partial S} \simeq \{x \in D : \bar{E}(x) > \alpha\}$ is effectively equivalent to empirical error $\bar{E}(x)$ exceeding α, so that the volume of $CI_{\partial S}$ is roughly proportional to the integrated empirical error $\mathcal{E}$.

In a more sophisticated approach based on random set theory, Chevalier et al. [24] used the

Vorob'ev deviation to define the uncertainty measure $V_\alpha(\hat{S})$:

$$\begin{aligned} V_\alpha(\hat{S}) :=& \mathbb{E}\left[\mu(\hat{S}^\alpha \Delta S)|\ \mathcal{A}\right] \\ =& \mathbb{E}\left[\int_{x\in D}\left\{\mathbb{I}[(x\in\hat{S}^\alpha \cap x\notin S)]\bigcup \mathbb{I}[(x\notin\hat{S}^\alpha \cap x\in S)]\right\}\mu(dx)\Big|\ \mathcal{A}\right] \\ =& \int_{\hat{S}^\alpha} p(x\notin S|\mathcal{A})\mu(dx) + \int_{(\hat{S}^\alpha)^C} p(x\in S|\mathcal{A})\mu(dx) \\ =& \int_{\hat{S}^\alpha}(1-p_+(x))\,\mu(dx) + \int_{(\hat{S}^\alpha)^C} p_+(x)\mu(dx), \end{aligned} \tag{3.39}$$

where

$$\hat{S}^\alpha := \left\{x\in D : \hat{f}(x) - z_{1-\frac{\alpha}{2}}s(x)\geq 0\right\} \quad \text{and} \quad p_+(x) = \mathbb{P}(x\in S|\mathcal{A}) = \Phi\left(\frac{\hat{f}(x)}{s(x)}\right).$$

An α satisfying the unbiasedness condition $\int_D p_+(x)\mu(dx) = \mathbb{E}[\mu(S)|\mathcal{A}] = \mu(\hat{S}^\alpha)$ is referred to as the *Vorob'ev threshold* and can be determined through dichotomy [24]. If the *Vorob'ev* threshold is picked to be zero, then the *Vorob'ev* deviation is reduced to the empirical error $\mathcal{E}$. Because of the computational overhead of working with (3.39), we restrict attention to the credible bands defined through $\hat{S}^\alpha$, which correspond to local uncertainty about f (or Z) as in (3.38).

Illustration. To visualize the performance statistics, we consider a one-dimensional case where we use the Gaussian observation GP to learn the sign of the quadratic $f(x) = x^2 - 0.75^2$ on $\mathcal{D} = [0,1]$, where $S = [0.75, 1]$ and with the unique zero contour at $x = 0.75$. The design $\mathbf{x}_{1:30}$ consists of $n = 30$ inputs drawn according to Latin hypercube sampling (LHS). The observations are $Y(x) = f(x) + \epsilon$, where $\epsilon \sim t_3(0, 0.1^2)$. In the top plot in Figure 3.1, we plot the true $f(\cdot)$, the posterior mean $\hat{f}^{(30)}(\cdot)$, and associated 95%-CI; in the respective bottom panel, we plot the local empirical error $\bar{E}$ as defined in (3.37). We also show the credible band for $\partial\hat{S}$ as defined in (3.38). Basically we observe a peak around the zero-contour $\partial S = 0.75$ for

local empirical error $\bar{E}$, indicating that this region has larger posterior probability of wrongly classifying x conditional on the training dataset $\mathcal{A}_{30}$. The global empirical error $\mathcal{E}$ defined in (3.36) is the area under the curve of the local empirical error $\bar{E}$. Volume of the credible interval $CI_{\partial S}$ is the length of the grey solid line, which is approximately the base of the triangle of the local empirical error cure. Beyond the endpoints of the credible interval $CI_{\partial S}$, the local empirical error $\bar{E}$ is close to zero. Besides, the maximum of $\bar{E}$ is always close to 0.5, where $\hat{f} \approxeq 0$, and thus $\bar{E} \approxeq 0.5$. Consequently the global empirical error $\mathcal{E}$ is approximately the area of the triangle with base equal to the volume of the credible interval $CI_{\partial S}$ and height equal to 0.5. So the global empirical error is approximately linear with the volume of the credible interval $CI_{\partial S}$, and we only provide measurements of the former one in our synthetic experiments. We also visualize the true zero-contour $\partial S = 0.75$ and its estimate $\partial \hat{S}$ in the bottom pannel of Figure 3.1. The distance between the blue circle and the red triangle is the error rate $\mathcal{ER}$ defined in (3.34).

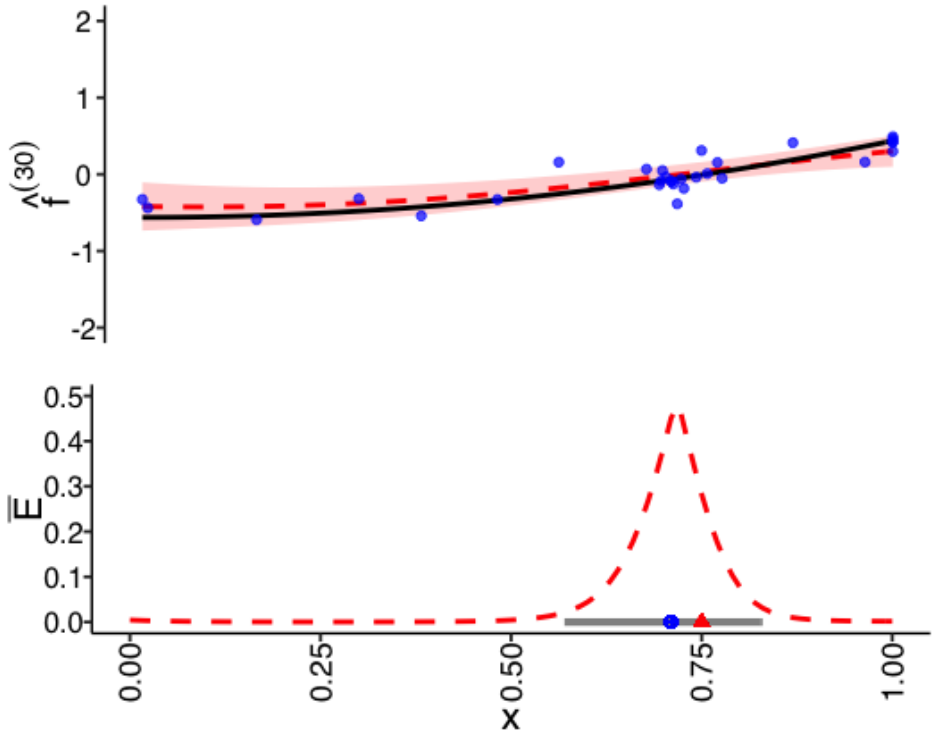


Figure 3.1: Visualization of performance statistics. *Upper* panel: true function $f = (x + 0.75)(x - 0.75)$ (black solid line), the posterior mean $\hat{f}(\cdot)$ (dashed line) and 95% CI_f (shaded area) based on observed samples $(\mathbf{x}_{1:30}, \mathbf{y}_{1:30})$ (blue dots). *Lower* panel: local empirical error $\bar{E}(x)$ defined in (3.37). Along the x-axis we also show the estimated zero-contour $\partial\hat{S}$ (blue circle) and its credible interval $CI_{\partial S}$ (grey solid line) relative the true zero-contour $\partial S = 0.75$ (red triangle).

Chapter 4

Sequential Design for Gaussian Process Surrogates

4.1 Sequential Design

We estimate the level set S in a sequential design setting that assumes that f is expensive to evaluate, for example because of the complexity of the underlying stochastic simulator. Therefore efficient selection of the inputs $\mathbf{x}_{1:n}$ is important. In sequential design, at each step the next sampling location x_{n+1} is selected given all previous measurements. The Bayesian approach to sequential design is based on greedily optimizing an acquisition function as in (2.7). These strategies got popularized thanks to the success of the expected improvement criterion and the associated efficient global optimization (EGO) algorithm [51]. The basic loop for sequential design is as following:

- Initialize $\mathcal{A}_{n_0} = \{(x_i, y_i), 1 \leq i \leq n_0\}$.
- Loop for $n = n_0 + 1, n_0 + 2, \ldots$.
  - Choose the next input $x_{n+1} = \arg\max_{x \in \mathcal{M}} \mathcal{I}_n(x)$, and sample $y_{n+1} = Y(x_{n+1})$.

- Augment $\mathcal{A}_{n+1} = \mathcal{A}_n \bigcup \{(x_{n+1}, y_{n+1})\}$.
- Update $\hat{S}^{(n+1)}$ with $\mathcal{A}_{n+1}$.

4.1.1 Literature Overview

Literature of sequential design mainly focuses on developing variants of acquisition function $\mathcal{I}_n(x)$ for different applications. Expected Improvement criterion [51] was one of the first sequential design strategies for Bayesian Optimization. It provides a measurement of expected maximum difference between the true minimum and the estimated function:

$$\mathcal{I}_n^{EI}(x) = \mathbb{E}[\max(y_{min} - f^{(n)}(x), 0)], \tag{4.1}$$

where $y_{min} = \min(y_1, ..., y_n)$. EI chooses the input whose expected response is the closest to the minimum observation.

Another important strategy for Bayesian Optimization, GP-UCB [97], chooses inputs with highest UCB and avoids regions where the UCB is smaller than that of the selected input :

$$\mathcal{I}_n^{UCB}(x) = \hat{f}^{(n)}(x) + \beta_n^{1/2} s^{(n)}(x). \tag{4.2}$$

Similar strategies were also investigated in [16, 38, 107] for level set estimation. Instead of the confidence bound of the response f, in level set estimation, our target is the zero-contour and its neighborhood, and thus we care about the confidence bound of its absolute value $|f|$. Bryan et al. [16] proposed straddle strategy:

$$\mathcal{I}_n^{STL}(x) = 1.96 s^{(n)}(x) - |\hat{f}^{(n)}(x)|. \tag{4.3}$$

The straddle strategy selects the input that is close to the zero-contour (small $|f|$) and have large

variance.

Bryan et al. [16] provided a complete comparison between several strategies for identifying the zero-contour in deterministic setting:

1. **Misclassification probability:** Choose the input which has the largest probability of being misclassified to the wrong level (i.e., $x \in S$ while $\hat{f}^{(n)} < 0$ or $x \in D$ while $\hat{f}^{(n)} > 0$) by the model. Let

$$\alpha^{(n)}(x) = \mathbb{P}(f^{(n)}(x) > 0) \tag{4.4}$$

be the probability that the response is greater than 0. Then the expected misclassification probability is defined as

$$\mathcal{I}_n^{MIS}(x) = \min(\alpha^{(n)}, 1 - \alpha^{(n)}). \tag{4.5}$$

Note equation (4.5) leads to the same expression as (3.37). However, (4.5) is used as an acquisition function in sequential design for deterministic experiments, while (3.37) is for stochastic experiments. Echard et al. [31, 32] explored the acquisition function

$$\mathcal{I}_n^{MCS}(x) = \frac{|\hat{f}^{(n)}(x)|}{s^{(n)}(x)}. \tag{4.6}$$

As an alternative criterion, MCS realizes the trade-off between the global uncertainty reduction (large $s^{(n)}$) and the exploration of the neighborhood of zero-contour (small $\hat{f}^{(n)}$).

2. **Entropy:** Choose the input which has the largest entropy

$$\mathcal{I}_n^{ETP}(x) = -\alpha^{(n)} \log_2(\alpha^{(n)}) - (1 - \alpha^{(n)}) \log_2(1 - \alpha^{(n)}). \tag{4.7}$$

Entropy is a monotone function of the misclassification rate so they will choose the same input as $\mathcal{I}_n^{MIS}(x)$.

3. **Variance:** Choose the input with large variance

$$\mathcal{I}_n^{VAR}(x) = s^{(n)}(x)^2. \tag{4.8}$$

It helps to explore the entire space and to avoid getting stuck in a cluster around one input or one contour.

4. **Information gain:** Choose the input with the largest information gain

$$\mathcal{I}_n^{IG}(x) = \mathcal{I}_n^{ETP}(x) - \mathbb{E}[\mathcal{I}_{n+1}^{ETP}(x)]. \tag{4.9}$$

Information gain is the same as entropy in deterministic setting, since the entropy at any input will be zero after one observation, leading to $\mathbb{E}[\mathcal{I}_{n+1}^{ETP}(x)]$ equal to zero. However, in stochastic setting, this value is the difference between entropy after one observation is simulated at the given input.

5. **Combination of metrics:** Choose the input with the largest combination of any two from the first four measures (4.5), (4.7), (4.8) and (4.9). Multiplying two measures together, say the entropy (4.7) and the variance (4.8), measures the classification uncertainty (entropy), as well as how much impact a new observation would have (variance). Another example is the staddle strategy, where we use the weighted sum of two measures together and strike a balance between them in sequential design.

Other level-set sampling criteria are based on modifications to the EI criterion which is originally designed for Bayesian Optimization. The Expected Feasibility Function [7] measures

how well the true value of the response is expected to be on the target zero-contour:

$$\mathcal{I}_n^{EFF}(x) = \int_{\epsilon}^{\epsilon} [\epsilon - |f^{(n)}(x)|] p(f^{(n)}(x)) df^{(n)}(x). \tag{4.10}$$

Similar to the EI criterion, EFF provides the same balance between exploration and exploitation. Inputs where the expected value is close to the zero-contour and inputs with a large uncertainty in the prediction will have large expected feasibility values. Ranjan et al. [89] proposed an expected improvement function defined for level set estimation and contour identification:

$$\mathcal{I}_n^{cEI}(x) = \mathbb{E}[(\alpha s^{(n)}(x))^2 - \min(f^{(n)}(x)^2, (\alpha s^{(n)}(x))^2)]. \tag{4.11}$$

The term $\alpha s^{(n)}(x)$ defines a neighborhood around the contour where the value of function is within $[-\alpha s^{(n)}(x), +\alpha s^{(n)}(x)]$. The criterion tends to be large if we sample from the zero-contour, where the predicted variance is largest.

The other criterion, targeted IMSE [83], aims at minimizing the IMSE in a target region instead of the entire input space. For level set estimation, the target region is a neighborhood around the zero-contour. With the zero contour, the targeted IMSE criterion is defined as

$$\begin{aligned} \mathcal{I}_n^{tIMSE}(x) &= \mathbb{E}[\int_{u \in D} s^{(n)}(u) \mathbb{I}[-\epsilon \leq f^{(n)}(u) \leq \epsilon] f^{(n)}(u) \mu(du) | x_{n+1} = x] \\ &= \int_{u \in D} s^{(n+1)}(u) W_n(u, x) \mu(du), \end{aligned} \tag{4.12}$$

where $W_n(\cdot)$ is a weight function. This criterion selects inputs to minimize the IMSE in a close neighborhood to the zero-contour. The weight function depends on value of ϵ, which controls the width of the close neighborhood around the zero-contour. For large ϵ, $\mathcal{I}_n^{tIMSE}$ performs approximately like the IMSE in surface metamodeling problem.

Stepwise Uncertainty Reduction is another category of strategies for level set estimation,

and it can be viewed as a global form of the local misclassification probability $\mathcal{I}_n^{MIS}$ (4.5). Define $\kappa_n(x)$ as the variance of the excess indicator $\mathbb{I}[f^{(n)}(x) > 0]$ obtained on n observations:

$$\kappa_n(x) = \alpha^{(n)}(x)(1 - \alpha^{(n)}(x)), \tag{4.13}$$

with $\alpha^{(n)}(\cdot)$ defined in (4.4). Several forms of SUR strategies were proposed based on global misclassification probability and global variance of $\mathbb{I}[f(x) > 0]$. In fact, $\mathcal{I}_n^{SUR-1}(x)$ and $\mathcal{I}_n^{SUR-3}(x)$ are global forms of local empirical error proposed in [16]. Note that $\mathcal{I}_n^{SUR-3}(x)$ for deterministic setup has the same expression as the empirical error $\mathcal{E}$ in (3.36) for stochastic experiments.

$\mathcal{I}_n^{SUR-1}(x) = \mathbb{E}[(\int \sqrt{\mathcal{I}_{n+1}^{MIS}(u)}\mu(du))^2 \vert x_{n+1} = x]$
$\mathcal{I}_n^{SUR-2}(x) = \mathbb{E}[(\int \sqrt{\kappa_{n+1}(u)}\mu(du))^2 \vert x_{n+1} = x]$
$\mathcal{I}_n^{SUR-3}(x) = \mathbb{E}[\int \mathcal{I}_{n+1}^{MIS}(u)\mu(du) \vert x_{n+1} = x]$
$\mathcal{I}_n^{SUR-4}(x) = \mathbb{E}[\int \kappa_{n+1}(u)\mu(du) \vert x_{n+1} = x]$

Table 4.1: SUR strategies proposed by Bect et al. [6].

4.1.2 Acquisition Functions for Level Set Estimation

We now propose several metrics for the acquisition function $\mathcal{I}_n(x)$ in Eq. (2.7). The key plan is to target regions close to the boundary $\partial\hat{S}$. A second strategy is to use the look-ahead posterior standard deviation $s^{(n+1)}$ conditional on sampling at x, in order to assess the corresponding *information gain*. This links the constructed design to the metamodel for f, since different surrogate architectures quantify uncertainty differently.

The first metric, dubbed Maximum Contour Uncertainty (MCU), stems from the GP-UCB strategies proposed by Srinivas et al. [97] for Bayesian Optimization. The idea of GP-UCB is to express the exploitation-exploration trade-off through the posterior mean $\hat{f}(x)$ and standard deviation $s(x)$. Following the spirit of GP-UCB, MCU blends the minimization of $|\hat{f}^{(n)}(x)|$

(exploitation) with maximization of the posterior uncertainty $s^{(n)}(x)$ (exploration):

$$\mathcal{I}_n^{\text{MCU}}(x) := -|\hat{f}^{(n)}(x)| + \gamma^{(n)} s^{(n)}(x), \tag{4.14}$$

where $\gamma^{(n)}$ is a step-dependent sequence of weights. Thus, MCU targets inputs with high uncertainty (large $s^{(n)}(x)$) and close to the boundary $\partial\hat{S}$ (small $|\hat{f}^{(n)}|$). Small $\gamma^{(n)}$ leads to aggressive sampling concentrated along the estimated $\partial\hat{S}$; large $\gamma^{(n)}$ leads to space-filling sampling that effectively minimizes the integrated mean-squared error. Thus, the choice of γ's is critical for the performance; in particular $\gamma^{(n)}$ should be increasing to avoid being trapped in local minima of $|\hat{f}^{(n)}(x)|$. In the original application to BO [97] it is proved that $\gamma^{(n)} = (2\log\left(\frac{|D|\pi^2 n^2}{6\delta}\right))^{1/2}$ is guaranteed to converge. Further recipes for choice of $\gamma^{(n)}$ in (4.14) for level set estimation were proposed in [38] and [11]; both papers mention that the above recommendation is too conservative and tends to over-explore. A constant choice of $\gamma^{(n)} = 1.96$ corresponds to the Straddle scheme in [16] and leads to $\mathcal{I}_n(x) \geq 0 \Leftrightarrow x \in$ (95% CI band of ∂S). Similarly, [38] employed $\gamma^{(n)} = 3$ and [11] suggested to use $\gamma^{(n)} = \sqrt{\log(|D|n^2)}$. Based on our experiments (see remark 1), we find that a constant value of $\gamma^{(n)}$ may be problematic and recommend to adapt $\gamma^{(n)}$ to the relative ratio between $f(x)$ (for steeper response surfaces γ should be larger) and $s(x)$ (γ needs to rise as posterior uncertainty decreases). One recipe is to use $\gamma^{(n)} = IQR(\hat{f}^{(n)})\backslash 3Ave(s^{(n)})$ which keeps both terms in (4.14) approximately comparable as n changes.

Remark 1 ***Choice of*** $\gamma^{(n)}$ ***for MCU*** *We investigate the role of* $\gamma^{(n)}$ *in the performance of MCU. Table 4.2 shows the error rate* $\mathcal{ER}$ *for GP and* t*-GP metamodels with MCU as acquisition function in the 2D synthetic experiments (see experiment setup in Section 4.3) with four noise structures across three constant values of* $\gamma^{(n)}$*. We observe that generally the impact of* $\gamma^{(n)}$ *is secondary (with Gaussian GP being more sensitive), and moreover there is no single choice that works the best across all cases. As illustrated in Figure 4.1, large* $\gamma^{(n)}$ *favors space-filling,*

while small $\gamma^{(n)}$ favors exploitation in regions close to the boundary ∂S.

Generally, smaller $\gamma^{(n)}$'s work better in cases with less noise (e.g. $\gamma^{(n)} = 10$ is worst in the t/small noise scenario). This validates our recommendation that $\gamma^{(n)}$ should be adaptive to the signal-to-noise ratio of $\hat{f}^{(n)}(x)$ and $s^{(n)}(x)$. Clearly $s^{(n)}(x)$ depends strongly on the original noise specification which is another reason why a fixed "universal" $\gamma^{(n)}$ is inappropriate (unlike in deterministic experiments, there is no simple way to normalize the variance of ϵ). Note that since $s^{(n)}$ decreases in n, the signal-to-noise ratio increases over time, which is consistent with the theoretical results that $\gamma^{(n)}$ should increase with n.

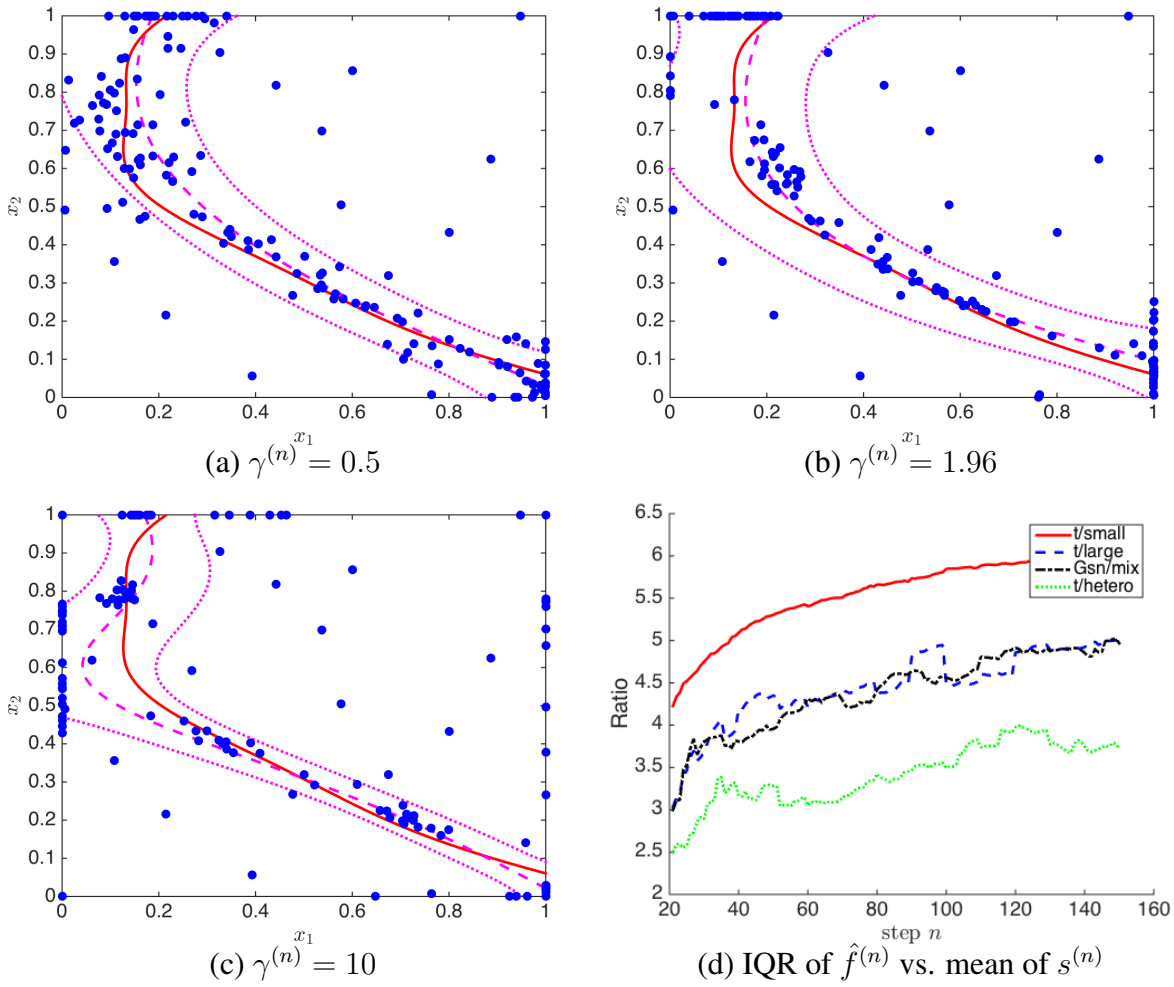
Model	$\gamma^{(n)} = 0.5$	$\gamma^{(n)} = 1.96$	$\gamma^{(n)} = 10$
		$\mathbf{t}$/small	
GP	1.87% (0.36%)	1.82% (0.51%)	2.09% (0.54%)
t-GP	1.80% (0.52%)	1.73% (0.22%)	1.84% (0.42%)
		$\mathbf{t}$/large	
GP	5.20% (2.33%)	5.59% (2.22%)	4.94% (1.79%)
t-GP	3.80% (1.25%)	4.24% (2.12%)	4.01% (1.43%)
		$\mathbf{Gsn}$/mix	
GP	5.10% (2.36%)	5.53% (1.79%)	6.01% (3.08%)
t-GP	4.63% (1.74%)	3.92% (1.26%)	4.39% (1.40%)
		$\mathbf{t}$/hetero	
GP	11.23% (5.08%)	10.52 % (7.05%)	13.63% (6.32%)
t-GP	7.34% (3.96%)	10.58% (8.25%)	7.77% (3.55%)

Table 4.2: Mean (w/standard deviation) error rate $\mathcal{ER}$ for MCU in 2D synthetic experiments with four noise structures. Results are based on 20 macro-replications of each scheme.

Remark 2 *The local empirical error $\bar{E}(x)$ as defined in Eq. (3.37) could be directly used as an acquisition function, i.e.*

$$\mathcal{I}_n^{MEE}(x) \equiv \bar{E}(x) = \Phi\bigg(- \frac{|\hat{f}^{(n)}(x)|}{s^{(n)}(x)} \bigg). \tag{4.15}$$

This Maximal Empirical Error (MEE) acquisition function measures the local probability of misclassification and is similar to the sequential criteria in [6, 31, 88, 7], all based on the idea of sampling at x where the event $\{f(x) > 0\}$ is most uncertain. However, (4.15) *is not suitable*



(a) $\gamma^{(n)} = 0.5$ (b) $\gamma^{(n)} = 1.96$

(c) $\gamma^{(n)} = 10$ (d) IQR of $\hat{f}^{(n)}$ vs. mean of $s^{(n)}$

Figure 4.1: (a)-(c): The estimated boundary $\partial\hat{S}$ (dashed line with 95% CI as dotted lines). Blue dots are samples selected by MCU with $\gamma^{(n)} = 0.5$, $\gamma^{(n)} = 1.96$, and $\gamma^{(n)} = 10$; (d) Ratio of IQR of $\hat{f}^{(n)}$ vs. mean of $s^{(n)}$ as a function of step n for 2D synthetic experiments with four noise structures: t/small (solid line), t/large (dash line), Gsn/mix (dash-dot line) and t/hetero (dot line). $\hat{f}^{(n)}$ and $s^{(n)}$ are calculated with GP. Samples are selected with MCU.

for our purposes since it is maximized across the entire $\partial\hat{S}$ *(namely* $\mathcal{I}_n^{MEE}(x) = 0.5$ *for any* x *where* $\hat{f}^{(n)}(x) = 0$*), so does not possess a unique maximizer as soon as* $\partial\hat{S}$ *is non-trivial. One potential solution could be to maximize* (4.15) *over a finite candidate set, which however requires significant fine-tuning.*

Our second strategy focuses on quickly *reducing* $\bar{E}$ by comparing the current $\bar{E}(x)$ given $\mathcal{A}_n$ and the expected $\bar{E}(x)$ conditional on the one-step-ahead sample, $\mathcal{A}_n \cup \{x_{n+1}, y_{n+1}\}$. This

is achieved by integrating out the effect of $Y(x_{n+1})$ on $\bar{E}(x_{n+1})$:

$$\mathcal{I}_n^{\text{cSUR}}(x) = \mathcal{I}_n^{\text{MEE}}(x) - \mathbb{E}_{Y(x)}\left[\mathcal{I}_{n+1}^{\text{MEE}}(x)\right] = \Phi\left(-\frac{|\hat{f}^{(n)}(x)|}{s^{(n)}(x)}\right) - \mathbb{E}_{Y(x)}\left[\Phi\left(-\frac{|\hat{f}^{(n+1)}(x)|}{s^{(n+1)}(x)}\right)\right]. \tag{4.16}$$

The name cSUR is because (4.16) is directly related to the SUR strategy [6], modified to target *c*ontour-finding. Crucially, $\mathcal{I}^{cSUR}$ ties the selection of x_{n+1} to the look-ahead mean $\hat{f}^{(n+1)}(x_{n+1})$ and look-ahead standard deviation $s^{(n+1)}(x_{n+1})$ that appear on the right-hand side of (4.17). To compute the integral over $Y(x)$, we replace $\hat{f}^{(n+1)}(x)$ with its average $\hat{f}^{(n)}(x) = \mathbb{E}_n[f(x)] = \mathbb{E}_n[\mathbb{E}_{n+1}[f(x)]] = \mathbb{E}_n[\hat{f}^{(n+1)}(x)]$. Similarly, we plug in the approximate one-step-ahead standard deviation $\hat{s}^{(n+1)}$ discussed in Section 4.2 (especially Equations (4.25), (4.38), and (4.40)) for $s^{(n+1)}(x)$:

$$\hat{\mathcal{I}}_n^{\text{cSUR}}(x) = \Phi\left(-\frac{|\hat{f}^{(n)}(x)|}{s^{(n)}(x)}\right) - \Phi\left(-\frac{|\hat{f}^{(n)}(x)|}{\hat{s}^{(n+1)}(x)|_{x_{n+1}=x}}\right). \tag{4.17}$$

Note that if x is such that $\hat{f}^{(n)}(x) = 0$ then both terms above are $1/2$ and $\mathcal{I}_n^{\text{cSUR}}(x) = 0$. Thus, the cSUR criterion will not place samples *directly* on $\partial\hat{S}$, but will aim to bracket the zero-contour.

In (4.17) cSUR only measures the *local* improvement in $\bar{E}(x_{n+1})$ at the sampling location x_{n+1} and consequently might be overly aggressive in targeting $\partial\hat{S}$. This motivates us to target the *global* reduction in the uncertainty of $\hat{S}$, so as to take into account the spatial structure of D. The resulting Integrated Contour Uncertainty (ICU) is linked to the already defined empirical error $\mathcal{E}$ from Section 3.6:

$$\mathcal{I}_n^{\text{ICU}}(x) := \mathbb{E}_{Y(x)}[\mathcal{E}^{(n+1)}|x_{n+1} = x] = \mathbb{E}_{Y(x)}\left[\int_{u\in D} \Phi\left(\frac{-|\hat{f}^{(n+1)}(u)|}{s^{(n+1)}(u)|_{x_{n+1}=x}}\right)\mu(du)\right]. \tag{4.18}$$

We apply the same approximation as for cSUR to simplify the expectation over $Y(x)$ and replace

the integral over D with a sum over a finite subset $\mathcal{D}$ of size M:

$$\hat{\mathcal{I}}_n^{\text{ICU}}(x) = -\sum_{x_m \in \mathcal{D}} \Phi\left(\frac{-|\hat{f}^{(n)}(x_m)|}{\hat{s}^{(n+1)}(x_m)|_{x_{n+1}=x}}\right)\mu(x_m). \tag{4.19}$$

Then $\mathcal{I}^{ICU}(x)$ can be viewed as measuring the overall information gain about S from sampling at x. The motivation behind ICU is to myopically minimize the expected one-step-ahead empirical error $\mathcal{E}$, which would correspond to 1-step Bayes-optimal design.

As a last alternative, we utilize the targeted mean square error (tMSE) criterion, a marginal form of targeted IMSE criterion in Picheny et al. [83]:

$$\mathcal{I}_n^{\text{tMSE}}(x) := s^{(n)}(x)^2 \cdot W_n^{\text{tMSE}}(x), \tag{4.20}$$

$$\text{where} \qquad W_n^{\text{tMSE}}(x) := \frac{1}{\sqrt{2\pi}s^{(n)}(x)} \exp\left(-\frac{\hat{f}_n(x)^2}{2s^{(n)}(x)^2}\right). \tag{4.21}$$

The tMSE criterion upweighs regions close to the zero contour through the weight function $W_n^{\text{tMSE}}(x)$ which measures the distance of x to $\partial\hat{S}^{(n)}$ using the Gaussian posterior density $\mathcal{N}(\hat{f}^{(n)}, s^{(n)}(x)^2)$. Like MCU, tMSE is based only on the posterior at step n and does not integrate over future $Y(x)$'s.

Remark 3 *In [83] an additional parameter σ_ϵ was added to the definition of $W_n^{tMSE}(x)$ by replacing $s^{(n)}(x)$ everywhere with $\sqrt{s^{(n)}(x)^2 + \sigma_\epsilon^2}$. We tried 0, 0.05, $s^{(n)}(x)$ and $1\backslash\sqrt{n}$ for σ_ϵ in* (4.21) *in 2D experiments to investigate the effect of value σ_ϵ on the performance of tMSE. The latter two are selected since in this way as the number of designs n increases, the target region gets narrower and exploration of the zero-contour is enhanced. Fitted plot is shown in Figure 4.2. We observe that, besides the last one, there is not a significant difference between the others in 2D experiments. We also repeated the experiments 20 times to account for randomness and the overall results remain the same. Therefore, for simplification, in this article, we use $\sigma_\epsilon = 0$. According to [83] σ_ϵ controls the size of the domain of interest of $\mathcal{I}_n^{tMSE}$: larger σ_ϵ yields more*

space-filling as $W_n^{tMSE}(x)$ *becomes flatter. Since [83] dealt with deterministic experiments,* σ_ϵ *is necessary to ensure that* $W_n^{tMSE}(x)$ *is well defined at existing* $x_{1:n}$ *and the recommendation was* σ_ϵ *to be 5% of the range of* f*. In our case* $s^{(n)}(x)$ *is intrinsically bounded away from zero and* (4.21) *works well as-is. Experiments indicate that the performance of* (4.20) *is not sensitive to* σ_ϵ*, so to minimize the number of tuning parameters we stick to* (4.21).

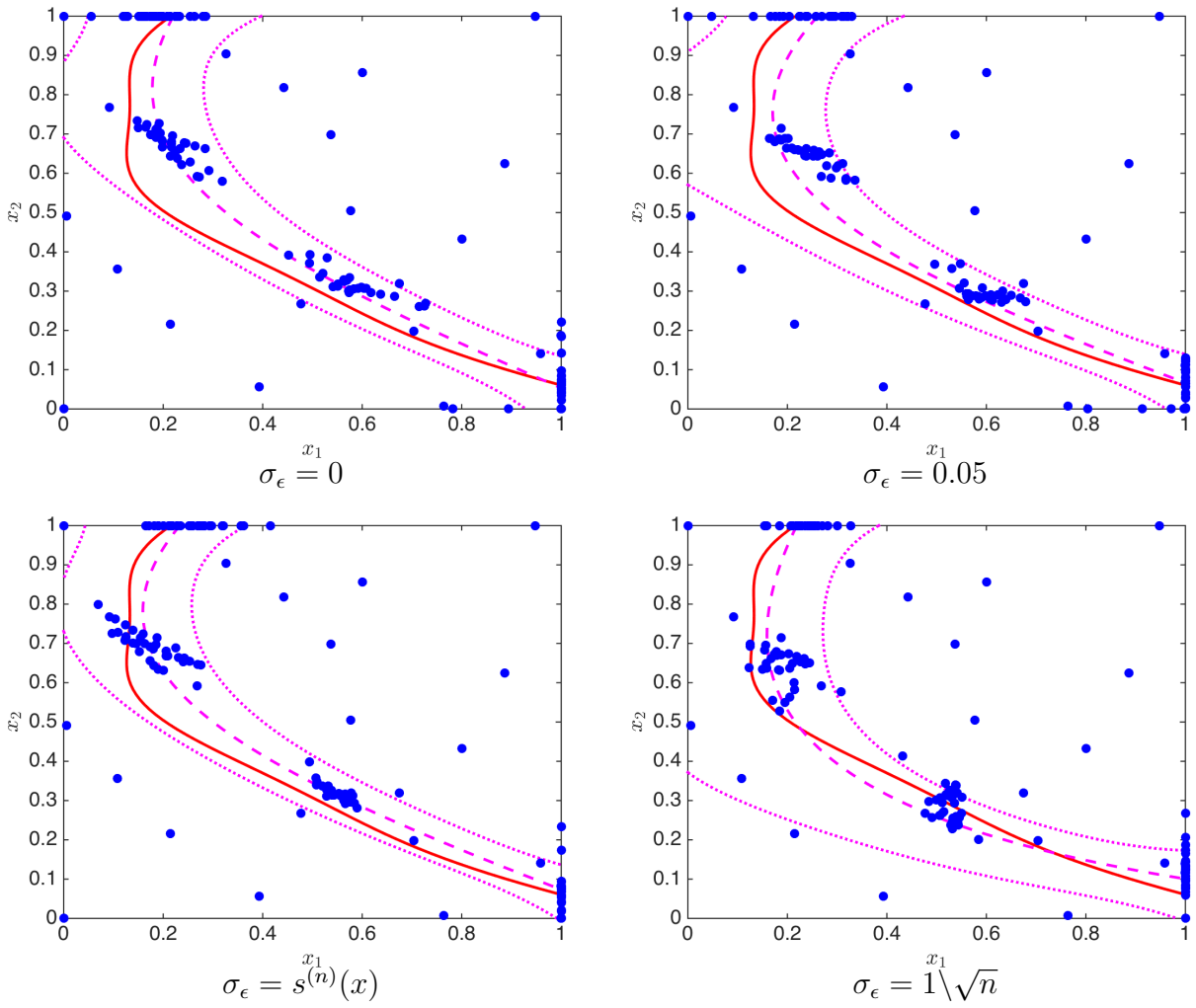


Figure 4.2: The estimated exercise boundary $\partial\hat{S}$ (dashed line with 95% CI as dotted lines). Blue dots are designs selected by tMSE with $\sigma_\epsilon = 0$, $\sigma_\epsilon = 0.05$, $\sigma_\epsilon = s^{(n)}(x)$ and $\sigma_\epsilon = 1\backslash\sqrt{n}$.

In the TP case, for MCU, cSUR, and ICU, we replace the standard normal cdf $\Phi(\cdot)$ appearing in the formulas by its Student-t counterpart (with the estimated degrees of freedom ν_n). For

tMSE, to maintain tractability, we keep the same expression (4.21) for the weights W^{tMSE}.

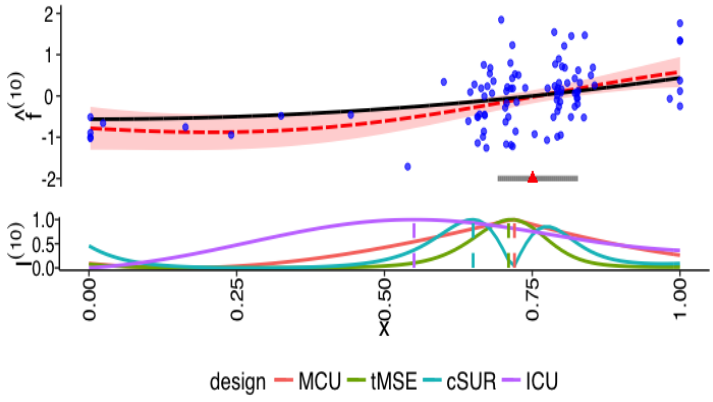


Figure 4.3: Comparison of acquisition functions. *Upper* panel: true function $f = (x + 0.75)(x - 0.75)$ (black solid line), the posterior mean $\hat{f}(\cdot)$ (dashed line) and 95% CI_f (shaded area) based on observed samples $(\mathbf{x}_{1:30}, \mathbf{y}_{1:30})$ (blue dots). Along the x-axis we also show the credible interval of the partition boundary $CI_{\partial S}$ (grey solid line) relative the true zero level set $S = [0, 0.75]$ (red triangle). *Lower* panel: acquisition functions $\mathcal{I}_n(\cdot)$ for MCU, cSUR, ICU, and tMSE criteria, with vertical lines marking the respective maxima $\arg\max_x \mathcal{I}_n(x)$.

Illustration

For instructive purposes, we reconsider the one-dimensional example discussed in Section 3.6. In the top plot in Figure 4.3, we plot the true $f(\cdot)$, the posterior mean $\hat{f}^{(30)}(\cdot)$, and associated 95%-CI. We also show the credible band for $\partial\hat{S}$; in the respective bottom panel, we plot the acquisition functions $\mathcal{I}_n^{\text{MCU}}(\cdot)$, $\mathcal{I}_n^{\text{cSUR}}(\cdot)$, $\mathcal{I}_n^{\text{ICU}}(\cdot)$ and $\mathcal{I}_n^{\text{tMSE}}(\cdot)$ as defined in Equations (4.14), (4.17), (4.19), and (4.20).

Comparing the acquisition functions of the four criteria, we find that, besides ICU, all of the others have maxima within the shaded credible interval of the boundary $CI_{\partial S}$. In practice, we care only about the maximizer of the acquisition function, rather than its full shape, since the former drives the selection of the next sample x_{n+1}. The x_{n+1}'s selected by MCU and tMSE criteria are close. For the cSUR criterion, because $\mathcal{I}_n^{\text{cSUR}}(x) = 0$ at $\partial\hat{S}$, there are two local maxima with a "valley" between them. The interval between the two local maxima is roughly

the confidence interval $CI_{\partial S}$ for the boundary (3.38). Both MCU and tMSE select a location very close to the boundary $\hat{f}^{(n)}(x_{n+1}) \simeq 0$. We note that MCU has a flatter acquisition function, i.e. tMSE is more aggressive. In contrast, the ICU and cSUR criteria are more "global"; in particular, ICU is the flattest among all the criteria.

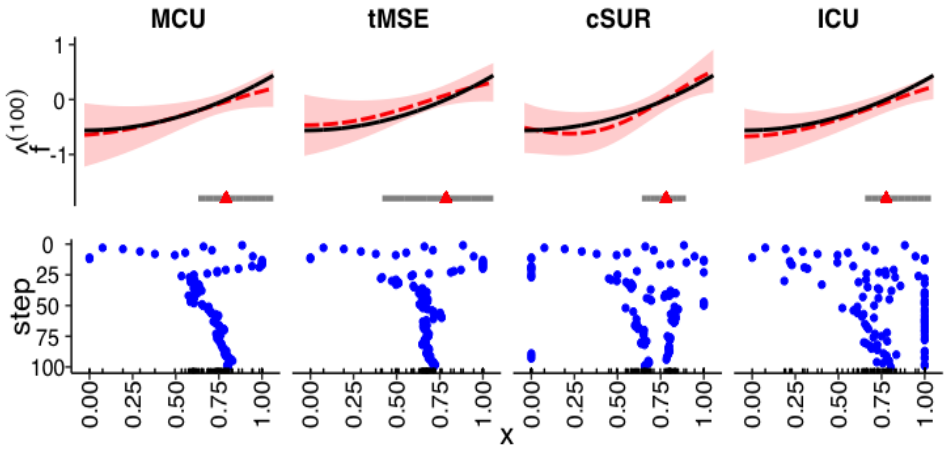


Figure 4.4: *Top row*: Fitted metamodel $\hat{f}^{(100)}$ (dashed red line) and its 95%-CI (shaded region) versus the true $f = (x + 0.75)(x - 0.75)$ (solid black), for each of the four design strategies. The estimated 95% CI for the zero-contour ∂S is marked on the x-axis with a grey interval; red triangle indicates the true zero-contour $\partial S = 0.75$. *Bottom row*: sampled inputs x_n (on the x-axis to match the top row) as a function of step $n = 1, \ldots, 100$ (on the y-axis, moving from top to bottom) for MCU, tMSE, cSUR, and ICU criteria. The rug plots at the bottom visualize the overall distribution of $\mathbf{x}_{1:n}$ at $n = 100$. The first ten inputs are selected using a (fixed-across schemes) LHS design on $D = [0, 1]$.

After using the various acquisition functions to select x_{n+1} at $n = 11, \ldots, 100$, we show in Figure 4.4 the resulting designs $\mathbf{x}_{1:n}$ and the final estimate $\hat{f}^{(100)}$ with a Gaussian observation GP metamodel. As desired, all methods target the true zero-contour at $\partial S = 0.75$. As a result, the posterior variance $s^{(n)}(x)^2$ is much lower in this neighborhood; in contrast, especially for tMSE and MCU, few samples are taken far from $x = 0.75$, and the posterior uncertainty there remains high. The true zero contour is within the estimated posterior CI for all the criteria. However, the CIs for MCU and tMSE are much wider than those for the others.

The bottom row in Figure 4.4 shows the sampled location x_n as a function of step n. We observe that MCU and tMSE heavily concentrate their search around the zero contour, leading

to few samples (and consequently relatively large empirical errors $\mathcal{E}^{(n)}$) in other areas, although the overall error rate $\mathcal{ER}$ is comparable. The ICU and cSUR criteria exhibit an "edge" effect; that is, besides the desired zero contour $x = 0.75$, multiple samples are taken close to the edges of the input space at $x = 0$ and $x = 1$. This occurs due to the relatively large posterior variance $s^2(\cdot)$ in those regions (which arises intrinsically with any spatial-based metamodel) that in turn strongly influences $\mathcal{I}^{\text{cSUR}}$ in (4.17) and $\mathcal{I}^{\text{ICU}}$ in (4.19). Inputs sampled by the cSUR criterion bracket the contour ∂S from both directions, matching the two-hill-and-a-valley shape of $\mathcal{I}^{\text{cSUR}}$ in Figure 4.3. We note that the two sampling "curves" get closer as n grows, indicating a gradual convergence of the estimated zero contour $\partial\hat{S}^{(n)}$, akin to a shrinking credible interval of $\hat{S}^{(n)}$. The ICU criterion generates a much more diffuse design: it engages in more exploration and is less dependent on the current levels of the empirical error $\mathcal{E}$. This eventually creates a flatter profile for $\bar{E}(x)$.

The preceding discussion considered a single metamodel choice for f. Other metamodels will generate different design features; in particular, sensitivity to $\epsilon(x)$ will lead to a different mix of exploration (x_n's far from the zero-contour) and exploitation even for the same choice of a $\mathcal{I}_n$ criterion. Figures 4.8 and 4.9, as well as Table 4.5, emphasize our message that one must jointly investigate the *combinations* of $\mathcal{I}(\cdot)$ and $\hat{f}$ when benchmarking the ultimate performance of the algorithm.

4.2 Look-Ahead Variance

The cSUR and ICU acquisition functions $\mathcal{I}_n$ require estimates of the look-ahead standard deviation $s^{(n+1)}(x_*)$ conditional on sampling at $x_{n+1} = x$. A related computation is also important for efficient updating of the GP/TP metamodels during sequential design, assimilating the observation (x_{n+1}, y_{n+1}) into $\mathcal{A}_n$. As is well known, usage of GP necessitates inverting the covariance matrix $\mathbf{K}^{-1}$ which presents a computational bottleneck as n grows. Updating hinges

on computing $(\mathbf{K}^{(n+1)})^{-1}$ via applying the Woodbury identities to the current $(\mathbf{K}^{(n)})^{-1}$.

A major advantage of the classical GP paradigm is that the posterior variance $s^{(n)}(x)^2$ is a function only of the design $\mathbf{x}_{1:n}$; that is, it is independent of the observations $\mathbf{y}_{1:n}$. This allows an exact analytic expression for $s^{(n+1)}(x)\big|_{x_{n+1}=x}$ in terms of x_{n+1}. Recall that for an existing design $\mathbf{x}_{1:n}$, after adding a new (x_{n+1}, y_{n+1}), the mean and variance at location x_* are updated via [25]

$$\hat{f}^{(n+1)}_{\mathrm{Gsn}}(x_*) = \hat{f}^{(n)}_{\mathrm{Gsn}}(x_*) + \lambda^{(n)}(x_*, x_{n+1})(y^{n+1} - \hat{f}^{(n)}_{\mathrm{Gsn}}(x_{n+1})) \tag{4.22}$$

$$s^{(n+1)}_{\mathrm{Gsn}}(x_*)^2 = s^{(n)}_{\mathrm{Gsn}}(x_*)^2 - \lambda^{(n)}(x_*, x_{n+1})^2(\tau^2 + s^{(n)}_{\mathrm{Gsn}}(x_{n+1})^2), \tag{4.23}$$

where $\lambda^{(n)}(x_*, x_{n+1})$ is a weight function that measures the influence of the new sample at x_{n+1} on x_* conditioned on the existing inputs $\mathbf{x}_{1:n}$.

Lemma 4.2.1 (Woodbury formula) *Assume* $\mathbf{b}$ *is a* $n \times 1$ *vector,* $\mathbf{A}$ *is a* $n \times n$ *matrix, and* d *and* c *are nonzero scalars; then we have*

$$[\mathbf{b}^T \quad d] \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^T & c \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b} \\ d \end{bmatrix} = \mathbf{b}^T\mathbf{A}^{-1}\mathbf{b} - \frac{1}{c - \mathbf{b}^T\mathbf{A}^{-1}\mathbf{b}}(d - \mathbf{b}^T\mathbf{A}^{-1}\mathbf{b})^2. \tag{4.24}$$

Using Lemma 4.2.1, we obtain the one-step-ahead variance at x_*:

Proposition 4.2.1 *For any* x_**,*

$$\lambda^{(n)}(x_*, x_{n+1}) = \frac{v^{(n)}_{\mathrm{Gsn}}(x_*, x_{n+1})}{\tau^2 + s^{(n)}_{\mathrm{Gsn}}(x_{n+1})^2} \quad \Rightarrow \quad s^{(n+1)}_{\mathrm{Gsn}}(x_*)^2 = s^{(n)}_{\mathrm{Gsn}}(x_*)^2 - \frac{v^{(n)}_{\mathrm{Gsn}}(x_*, x_{n+1})^2}{\tau^2 + s^{(n)}_{\mathrm{Gsn}}(x_{n+1})^2}. \tag{4.25}$$

In particular, after sampling at x_{n+1} *the local updated posterior variance is proportional to the*

current $s^{(n)}_{\mathrm{Gsn}}(x_{n+1})^2$ *with a proportionality factor [44]:*

$$\frac{s^{(n+1)}_{\mathrm{Gsn}}(x_{n+1})^2}{s^{(n)}_{\mathrm{Gsn}}(x_{n+1})^2} = \frac{\tau^2}{\tau^2 + s^{(n)}_{\mathrm{Gsn}}(x_{n+1})^2}. \tag{4.26}$$

The above lemma is our basis for calculating the acquisition function for the cSUR criterion (4.17) that requires only (4.26) and the ICU criterion (4.19). As we see below, because (4.25) holds only in the Gaussian prior/Gaussian likelihood setting, further approximations are required to apply (4.2.1)–(4.26) for the alternative metamodels. Such look-ahead variance expressions are of independent interest, applicable beyond the context of level set estimation.

A limitation of using a non-Gaussian observation or classification likelihood is that, unlike for Gaussian observation GP, there are no exact variance look-ahead formulas for the resulting t-GP, Cl-GP and TP metamodels. There are two main reasons for this. First, both the posterior mean $\hat{f}^{(n+1)}(x_*)$ in (3.13) and (3.19) and the posterior variance $s^{(n+1)}(x_*)^2$ in (3.14) and (3.20) for t-GP and Cl-GP depend on the posterior mode $\tilde{\mathbf{f}}^{(n+1)}_{t\mathrm{GP}}$ or $\tilde{\mathbf{z}}^{(n+1)}_{Cl}$, which changes every step. Therefore, they cannot be accessed in advance. Furthermore, for t-GP, Cl-GP and M-GP $s^{(n+1)}(x_*)$ depends on the next-step Hessian $\boldsymbol{W}$ (namely on $w^{(n+1)}_{n+1}$), and for TP $s^{(n+1)}(x_*)$ depends on $\beta^{(n+1)}$. Both of these again depend on y_{n+1}. To overcome this challenge, we develop an approximation $\hat{s}^{(n+1)}(\cdot)$ for each metamodel. Our strategy is to replace each inaccessible term with its expected value from the point of view of step n. For example, we calculate the expectation of $\tilde{\mathbf{f}}^{(n+1)}_{t\mathrm{GP}}$, $\tilde{\mathbf{z}}^{(n+1)}_{Cl}$, $\tilde{\boldsymbol{\Sigma}}^{(n+1)}_{Mon}$ and $\beta^{(n+1)}$ with respect to $\mathcal{A}_n$. Propositions 4.2.2-4.2.3-4.2.5 provide the resulting look-ahead formulas for t-GP, Cl-GP, M-GP and TP respectively, with derivation details in Section 4.2.1.

Proposition 4.2.2 *For any* x_**, the formula for the look-ahead variance for* t*-GP is*

$$\hat{s}^{(n+1)}_{t\mathrm{GP}}(x_*)^2 := s^{(n)}_{t\mathrm{GP}}(x_*)^2 - \frac{v^{(n)}_{t\mathrm{GP}}(x_*, x_{n+1})^2}{(\tau^2 \frac{\nu+1}{\nu-1}) + s^{(n)}_{t\mathrm{GP}}(x_{n+1})^2}. \tag{4.27}$$

Proposition 4.2.3 *Let* $\check{v}_{n+1} = v^+_{n+1}p_+ + v^-_{n+1}p_-$, *where*

$$\textit{where} \qquad v^+_{n+1} = \frac{\phi(\hat{z}^{(n)}_{\mathrm{Cl}}(x_{n+1}))^2}{\Phi(\hat{z}^{(n)}_{\mathrm{Cl}}(x_{n+1}))^2} + \frac{\hat{z}^{(n)}_{\mathrm{Cl}}(x_{n+1})\phi(\hat{z}^{(n)}_{\mathrm{Cl}}(x_{n+1}))}{\Phi(\hat{z}^{(n)}_{\mathrm{Cl}}(x_{n+1}))}, \tag{4.28}$$

$$p_+ = \Phi\left(\frac{\hat{z}^{(n)}(x_{n+1})}{\sqrt{1 + s^{(n)}_C(x_{n+1})^2}}\right), \tag{4.29}$$

$$\textit{and} \qquad v^-_{n+1} = \frac{\phi(\hat{z}^{(n)}_{\mathrm{Cl}}(x_{n+1}))^2}{\Phi(-\hat{z}^{(n)}_{\mathrm{Cl}}(x_{n+1}))^2} - \frac{\hat{z}^{(n)}_{\mathrm{Cl}}(x_{n+1})\phi(\hat{z}^{(n)}_{\mathrm{Cl}}(x_{n+1}))}{\Phi(-\hat{z}^{(n)}_{\mathrm{Cl}}(x_{n+1}))}, \quad p_- = 1 - p_+. \tag{4.30}$$

For any x_*, *the formula for the look-ahead variance for Cl-GP is*

$$\hat{s}^{(n+1)}_{\mathrm{Cl}}(x_*)^2 := s^{(n)}_{\mathrm{Cl}}(x_*)^2 - \frac{v^{(n)}_{\mathrm{Cl}}(x_*, x_{n+1})^2}{(\check{v}_{n+1})^{-1} + s^{(n)}_{\mathrm{Cl}}(x_{n+1})^2}. \tag{4.31}$$

Proposition 4.2.4 *For monotonic GP, the local updated variance* $s^{(n+1)}_{\mathrm{Mon}}(x_{n+1})^2$ *at* x_{n+1}, *and the step-ahead variance* $s^{(n+1)}_{\mathrm{Mon}}(x_*)^2$ *at any input* x_*:

$$\frac{s^{(n+1)}_{\mathrm{Mon}}(x_{n+1})^2}{s^{(n)}_{\mathrm{Mon}}(x_{n+1})^2} = \frac{\tau^2}{\tau^2 + s^{(n)}_{\mathrm{Mon}}(x_{n+1})^2}, \tag{4.32}$$

$$s^{(n+1)}_{\mathrm{Mon}}(x_*)^2 = s^{(n)}_{\mathrm{Mon}}(x_*)^2 - \frac{v^{(n)}_{\mathrm{Mon}}(x_*, x_{n+1})^2}{\tau^2 + s^{(n)}_{\mathrm{Mon}}(x_{n+1})^2}. \tag{4.33}$$

Proposition 4.2.5 *For any* x_*, *the formula for the look-ahead variance for TP is*

$$s^{(n+1)}_{\mathrm{TP}}(x_*)^2 = \frac{\nu + \check{\beta}^{(n+1)} - 2}{\nu + n - 1} s^{(n+1)}_{\mathrm{Gsn}}(x_*)^2, \tag{4.34}$$

where

$$\check{\beta}^{(n+1)} = \beta^{(n)} + \frac{\nu}{\nu - 2}.$$

We note that in our experiments we only use the above to evaluate $\mathcal{I}_n$, and directly re-estimate $\tilde{f}^{(n+1)}$ at each step of the sequential design.

4.2.1 Computation Details for Look-Ahead Variance

t-GP: To approximate $\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n+1)}$ in t-GP, we recall that the posterior mode and the posterior mean coincide:

$$\hat{\mathbf{f}}_{t\mathrm{GP}}^{(n)}(\mathbf{x}_{1:n}) = \mathbf{K}\mathbf{K}^{-1}\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n)} = \tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n)}. \tag{4.35}$$

Hence we can compute the expected value of $\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n+1)}$ using the tower property:

$$\begin{aligned}\mathbb{E}[\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n+1)}|\mathbf{x}_{1:n},\mathbf{y}_{1:n}] &= \mathbb{E}[\hat{\mathbf{f}}_{t\mathrm{GP}}^{(n+1)}(\mathbf{x}_{1:n+1})|\mathbf{x}_{1:n},\mathbf{y}_{1:n}] = \mathbb{E}\left[\mathbb{E}[f(\mathbf{x}_{1:n+1})|\mathbf{x}_{1:n+1},\mathbf{y}_{1:n+1}]|\mathbf{x}_{1:n},\mathbf{y}_{1:n}\right] \\ &= \mathbb{E}[f(\mathbf{x}_{1:n+1})|\mathbf{x}_{1:n},\mathbf{y}_{1:n}] = [\hat{\mathbf{f}}_{t\mathrm{GP}}^{(n)}(\mathbf{x}_{1:n}), \hat{f}_{t\mathrm{GP}}^{(n)}(x_{n+1})] = [\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n)}, \hat{f}_{t\mathrm{GP}}^{(n)}(x_{n+1})],\end{aligned} \tag{4.36}$$

where the last equality follows from the BLUP property of GP estimates. Therefore, we approximate the $(n+1)$-dimensional vector $\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n+1)}$ with $\check{\mathbf{f}}_{t\mathrm{GP}}^{(n+1)} = [\tilde{\mathbf{f}}_{t\mathrm{GP}}^{(n)}, \hat{f}_{t\mathrm{GP}}^{(n)}(x_{n+1})]$, where the first component is n-dimensional and the second component is a scalar. In turn, this step allows us to update the matrices $\boldsymbol{W}_{t\mathrm{GP}}^{(n)}$ and $\boldsymbol{K}^{(n)}$ assuming a new input x_{n+1} is added. Specifically, the new entry in $\boldsymbol{W}_{t\mathrm{GP}}^{(n+1)}$ is

$$\begin{aligned}w_{n+1}^{(n+1)} &= (\nu+1)\frac{\nu\tau^2 - (y_{n+1} - \tilde{f}_{t\mathrm{GP}}^{(n+1)}(x_{n+1}))^2}{\left((y_{n+1} - \tilde{f}_{t\mathrm{GP}}^{(n+1)}(x_{n+1}))^2 + \nu\tau^2\right)^2} \\ &\simeq (\nu+1)\frac{\nu\tau^2 - (y_{n+1} - \hat{f}_{t\mathrm{GP}}^{(n)}(x_{n+1}))^2}{\left((y_{n+1} - \hat{f}_{t\mathrm{GP}}^{(n)}(x_{n+1}))^2 + \nu\tau^2\right)^2}.\end{aligned} \tag{4.37}$$

Matching terms with the Gaussian observation GP, the updated variance $s_{t\mathrm{GP}}^{(n+1)}(x_{n+1})^2$ is then approximately proportional to the current variance:

$$\frac{s_{t\mathrm{GP}}^{(n+1)}(x_{n+1})^2}{s_{t\mathrm{GP}}^{(n)}(x_{n+1})^2} \simeq \frac{(w_{n+1}^{(n+1)})^{-1}}{(w_{n+1}^{(n+1)})^{-1} + s_{t\mathrm{GP}}^{(n)}(x_{n+1})^2}. \tag{4.38}$$

To make this implementable at step n, we need to remove the inaccessible y_{n+1} term in both the numerator and denominator of (4.37). In principle, we could attempt to (numerically) integrate the predictive distribution $Y(x_{n+1}) \sim t_\nu(f_t^{(n)}(x_{n+1}), \tau^2)$ against $f_t^{(n)}(x_{n+1}) \sim \mathcal{N}(\hat{f}_t^{(n)}(x_{n+1}), s_t^{(n)}(x_{n+1})^2)$; for simplicity we instead replace $(y_{n+1} - \hat{f}_{t\mathrm{GP}}^{(n)}(x_{n+1}))^2$ with its expectation: $\mathbb{E}[(y_{n+1} - \hat{f}_{t\mathrm{GP}}^{(n)}(x_{n+1}))^2] = \mathrm{Var}[y_{n+1}] = \tau^2$ and therefore obtain the approximation $w_{n+1}^{(n+1)} \simeq (\nu+1)\frac{(\nu-1)\tau^2}{(\nu+1)^2\tau^4} = \frac{\nu-1}{(\nu+1)\tau^2}$. This leads to the final look-ahead variance formula (cf. (4.25)):

$$\hat{s}_{t\mathrm{GP}}^{(n+1)}(x_*; x_{n+1})^2 := s_{t\mathrm{GP}}^{(n)}(x_*)^2 - \frac{v_{t\mathrm{GP}}^{(n)}(x_*, x_{n+1})^2}{(\tau^2 \frac{\nu+1}{\nu-1}) + s_{t\mathrm{GP}}^{(n)}(x_{n+1})^2}. \tag{4.39}$$

Cl-GP: Similar to the t-GP, the look-ahead variance for the classification GP is intractable since $s_{\mathrm{Cl}}^{(n+1)}$ is based on the mode $\tilde{\mathbf{z}}_{\mathrm{Cl}}^{(n+1)}$ of the posterior $p_{\mathrm{Cl}}(\mathbf{z}|\mathbf{x}_{1:n}, \mathbf{y}_{1:n}, x_{n+1}, y_{n+1})$. Similar to (4.36) we use the approximation $\tilde{\mathbf{z}}_{\mathrm{Cl}}^{(n+1)} \simeq \check{\mathbf{z}}_{\mathrm{Cl}}^{(n+1)} := [\tilde{\mathbf{z}}_{\mathrm{Cl}}^{(n)}, \hat{z}_{\mathrm{Cl}}^{(n)}(x_{n+1})]$. In that case we obtain an expression similar to (4.38), with $w_{n+1}^{(n+1)}$ replaced by $v_{n+1}^{(n+1)}$ from Eq. (3.18):

$$\frac{s_{\mathrm{Cl}}^{(n+1)}(x_{n+1})^2}{s_{\mathrm{Cl}}^{(n)}(x_{n+1})^2} \simeq \frac{(v_{n+1}^{(n+1)})^{-1}}{(v_{n+1}^{(n+1)})^{-1} + s_{\mathrm{Cl}}^{(n)}(x_{n+1})^2}. \tag{4.40}$$

The Hessian element $v_{n+1}^{(n+1)}$ is given by

$$v_{n+1}^{(n+1)} = \frac{\phi(\tilde{z}_{n+1}^{(n+1)})^2}{\Phi(\check{y}_{n+1}\tilde{z}_{n+1}^{(n+1)})^2} + \frac{\check{y}_{n+1}\tilde{z}_{n+1}^{(n+1)}\phi(\tilde{z}_{n+1}^{(n+1)})}{\Phi(\check{y}_{n+1}\tilde{z}_{n+1}^{(n+1)})},$$

which depends on the next-step signed response $\check{y}_{n+1}$. To develop an approximation in terms of step-n values, we once more replace $\tilde{z}_{n+1}^{(n+1)}$ with the current mean $\hat{z}_{\mathrm{Cl}}^{(n)}(x_{n+1})$. Moreover, the next response $\check{y}_{n+1}$ will take only two values, so $v_{n+1}^{(n+1)}$ will take on just two values $v_{n+1}^{\pm}$. Hence,

we can compute the "expected value"

$$\check{v}_{n+1} := v^+_{n+1} p_+ + v^-_{n+1} p_-, \tag{4.41}$$

$$\text{where} \quad v^+_{n+1} = \frac{\phi(\hat{z}^{(n)}_{\text{Cl}}(x_{n+1}))^2}{\Phi(\hat{z}^{(n)}_{\text{Cl}}(x_{n+1}))^2} + \frac{\hat{z}^{(n)}_{\text{Cl}}(x_{n+1})\phi(\hat{z}^{(n)}_{\text{Cl}}(x_{n+1}))}{\Phi(\hat{z}^{(n)}_{\text{Cl}}(x_{n+1}))}, \tag{4.42}$$

$$\text{and} \quad v^-_{n+1} = \frac{\phi(\hat{z}^{(n)}_{\text{Cl}}(x_{n+1}))^2}{\Phi(-\hat{z}^{(n)}_{\text{Cl}}(x_{n+1}))^2} - \frac{\hat{z}^{(n)}_{\text{Cl}}(x_{n+1})\phi(\hat{z}^{(n)}_{\text{Cl}}(x_{n+1}))}{\Phi(-\hat{z}^{(n)}_{\text{Cl}}(x_{n+1}))}, \tag{4.43}$$

with $p_+ := \mathbb{P}(Y(x_{n+1}) > 0|\mathcal{A}_n) = \int_{\mathbb{R}} \Phi(z) p_{Z(x_{n+1})}(z|\mathcal{A}_n)dz = \Phi\left(\frac{\hat{z}^{(n)}(x_{n+1})}{\sqrt{1+s^{(n)}_C(x_{n+1})^2}}\right)$, and $p_- = 1 - p_+$. The final formula for the look-ahead variance becomes

$$\hat{s}^{(n+1)}_{\text{Cl}}(x_{n+1})^2 := s^{(n)}_{\text{Cl}}(x_{n+1})^2 \cdot \frac{(\check{v}_{n+1})^{-1}}{(\check{v}_{n+1})^{-1} + s^{(n)}_{\text{Cl}}(x_{n+1})^2}.$$

Monotonic GP: Similar to the t-GP and Cl-GP, look-ahead variance is intractable for the monotonic GP, since the EP mean $\tilde{\boldsymbol{\mu}}_{\text{Mon}}$ and variance $\tilde{\boldsymbol{\Sigma}}_{\text{Mon}}$ are changing as the designs are augmented. Rewriting (3.30), we obtain

$$\tilde{v}^{(n)}_{\text{Mon}}(x_*, x'_*) = K_{\mathbf{f},\mathbf{f}}(x_*, x'_*) - [K^{(n)}_{\mathbf{f},\mathbf{f}'}(x_*, \mathbf{x}_v), k(x_*)](\tilde{\mathbf{K}}^{(n)}_{\text{joint}} + \tilde{\tilde{\boldsymbol{\Sigma}}}^{(n)}_{\text{joint}})^{-1} \begin{bmatrix} K^{(n)}_{\mathbf{f}',\mathbf{f}}(\mathbf{x}_v, x_*) \\ k(x_*) \end{bmatrix}, \tag{4.44}$$

$$\text{where} \quad \tilde{\mathbf{K}}^{(n)}_{\text{joint}} = \begin{bmatrix} K^{(n)}_{\mathbf{f}',\mathbf{f}'}(\mathbf{x}_v, \mathbf{x}_v) & K^{(n)}_{\mathbf{f}',\mathbf{f}}(\mathbf{x}_v, \mathbf{x}_{1:n}) \\ K^{(n)}_{\mathbf{f},\mathbf{f}'}(\mathbf{x}_{1:n}, \mathbf{x}_v) & K^{(n)}_{\mathbf{f},\mathbf{f}}(\mathbf{x}_{1:n}, \mathbf{x}_{1:n}) \end{bmatrix} \quad \text{and} \quad \tilde{\tilde{\boldsymbol{\Sigma}}}^{(n)}_{\text{joint}} = \begin{bmatrix} \tilde{\boldsymbol{\Sigma}}^{(n)}_{\text{Mon}} & \mathbf{0} \\ \mathbf{0} & \sigma^2 \mathbf{I}_{n\times n} \end{bmatrix}.$$

$\boldsymbol{K}^{(n)}_{\mathbf{f}',\mathbf{f}'}(\mathbf{x}_v, \mathbf{x}_v)$ is the step-n covariance matrix for the gradient of virtual observations, and $\tilde{\boldsymbol{\Sigma}}^{(n)}_{\text{Mon}}$ is the approximate covariance matrix for $p_{\text{Mon}}(\mathbf{y}_v|\mathbf{f}')$. When calculating the one-step-ahead variance for monotonic GP, we freeze the virtual observations and their gradient, which in consequence freezes the $K^{(n)}_{\mathbf{f}',\mathbf{f}'}(\mathbf{x}_v, \mathbf{x}_v)$, $K^{(n)}_{\mathbf{f}',\mathbf{f}}(\mathbf{x}_v, \mathbf{x}_{1:n})$, $K^{(n)}_{\mathbf{f},\mathbf{f}'}(\mathbf{x}_{1:n}, \mathbf{x}_v)$, and $\tilde{\boldsymbol{\Sigma}}^{(n)}_{\text{Mon}}$ matrices.

Therefore, the virtual observations are treated as fixed inputs. Then, as a new observation is added, only the last row and column of the covariance matrix are updated, while the other parts remain unchanged. This approach transforms computing the look-ahead standard deviation $s^{(n+1)}_{\text{Mon}}$ into the classical Gaussian observation GP as in (4.26).

TP: In terms of update formulas, TPs are in between GPs and t-GPs, with closed-form expressions available but depending on y_{n+1}. Specifically, the effect of adding a new observation (x_{n+1}, y_{n+1}) can be highlighted in closed form, since

$$f(x_*)|\mathbf{y}_{1:n}, y_{n+1} \sim \mathcal{T}\left(\nu + n + 1, \hat{f}^{(n+1)}_{\text{TP}}(x_*), s^{(n+1)}_{\text{TP}}(x_*)\right),$$

where

$$\hat{f}^{(n+1)}_{\text{TP}}(x_*) = \hat{f}^{(n+1)}_{\text{Gsn}}(x_*) \tag{4.45}$$

$$s^{(n+1)}_{\text{TP}}(x_*)^2 = \frac{\nu + \beta^{(n+1)} - 2}{\nu + n - 1} s^{(n+1)}_{\text{Gsn}}(x_*)^2. \tag{4.46}$$

The effect of y_{n+1} is inside

$$\begin{aligned}\beta^{(n+1)} &= \mathbf{y}^\top_{1:n+1}(\mathbf{K}^{(n+1)})^{-1}\mathbf{y}_{1:n+1} \\ &= \beta^{(n)} + s^{(n)}_{\text{Gsn}}(x_{k_n+1})^{-1}\left(h^{(n)}(x_{k_n+1})^2 + 2y_{n+1}h^{(n)}(x_{k_n+1}) + y^2_{n+1}\right),\end{aligned}$$

using the partition inverse equation, with $h^{(n)}(x) := -\mathbf{y}^\top_{1:n}(\mathbf{K}^{(n)})^{-1}k(x) = -\hat{f}_{\text{Gsn}}(x)$. Since y_{n+1} is unknown beforehand, we use a plugin value $\check{\beta}^{(n+1)}$ for $\beta^{(n+1)}$, relying again on the tower

property:

$$\begin{aligned}\check{\beta}^{(n+1)} &= \mathbb{E}[\beta^{(n+1)}|\mathbf{x}_{1:n}, \mathbf{y}_{1:n}] = \mathbb{E}\left[\mathbb{E}[\beta^{(n+1)}|\mathbf{x}_{1:n+1}, \mathbf{y}_{1:n+1}]|\mathbf{x}_{1:n}, \mathbf{y}_{1:n}\right] \\ &= \beta^{(n)} + s^{(n)}_{\mathrm{Gsn}}(x_{k_n+1})^{-2} \\ &\left(h^{(n)}(x_{k_n+1})^2 + 2\hat{f}^{(n)}_{\mathrm{Gsn}}(x_{k_n+1})h^{(n)}(x_{k_n+1}) + \hat{f}^{(n)}_{\mathrm{Gsn}}(x_{k_n+1})^2 + \frac{\nu}{\nu-2}s^{(n)}_{\mathrm{Gsn}}(x_{k_n+1})^2\right) \\ &= \beta^{(n)} + \frac{\nu}{\nu-2}. \end{aligned} \tag{4.47}$$

4.3 Synthetic Experiments

4.3.1 Benchmark Construction

As synthetic experiments, we consider three benchmark problems in dimension $d = 1, 2$, and 6. For the latter two we employ the widely used *Branin-Hoo* 2-D and *Hartman* 6-D functions; see, for example, [84]. The original functions have been rescaled to map their sample space D onto $[0, 1]^d$; see Table 4.3.

The latent functions are chosen to cover a variety of problem properties. The quadratic f in 1-D is strictly monotonically increasing, yielding a single boundary ∂S. The original Branin-Hoo function [84] is modified so that f is increasing in x^1 and the zero-level set has a non-trivial shape in x^2. The *Hartman* is a multimodal function with a complex zero contour. The parameters in the original *Hartman* function described in [84] are adjusted to reduce the "bumps" in the zero contour and make the problem more appropriate for the level set estimation task.

A large number of factors can influence the performance of metamodels and designs. In line with the stochastic simulation perspective, we concentrate on the impact of the simulation noise and consider four observation setups. These cover a variety of noise distributions and signal-to-noise ratio, measured through the proportion of standard deviation τ to the range R_f

Quadratic (1-D)	$f(x) = (x + 0.75)(x - 0.75)$ with $x \in [0, 1]$
Branin-Hoo (2-D)	$f(x) = \frac{1}{178}\left[\left(\bar{x}^1 - \frac{5.1(\bar{x}^2)^2}{4\pi^2} + \frac{5\bar{x}^2}{\pi} - 20\right)^2 + (10 - \frac{10}{8\pi})\cos(\bar{x}^1) - 181.47\right]$ with: $\bar{x}^1 = 15x^1$, $\bar{x}^2 = 15x^2 - 5$, $x^1, x^2 \in [0, 1]$
Hartman6 (6-D)	$f(x) = \frac{-1}{0.1}\left[\sum_{i=1}^4 C_i \exp\left(-\sum_{j=1}^6 a_{ji}(x^j - p_{ji})^2\right) - 0.1\right]$ with: $\mathbf{C} = [0.2, 0.22, 0.28, 0.3]$ $\mathbf{a} = \begin{bmatrix} 8.00 & 0.50 & 3.00 & 10.00 \\ 3.00 & 8.00 & 3.50 & 6.00 \\ 10.00 & 10.00 & 1.70 & 0.50 \\ 3.50 & 1.00 & 8.00 & 8.00 \\ 1.70 & 6.00 & 10.00 & 1.00 \\ 6.00 & 9.00 & 6.00 & 9.00 \end{bmatrix}$ $\mathbf{p} = \frac{1}{10^4}\begin{bmatrix} 1312 & 2329 & 2348 & 4047 \\ 1696 & 4135 & 1451 & 8828 \\ 5569 & 8307 & 3522 & 8732 \\ 124 & 3736 & 2883 & 5743 \\ 8283 & 1004 & 3047 & 1091 \\ 5886 & 9991 & 6650 & 381 \end{bmatrix}$

Table 4.3: Response surfaces $x \mapsto f(x)$ for synthetic experiments.

of the response. The first two settings use *Student-t* distributed noise, with (i) low τ and (ii) high τ. The third setting uses (iii) Gaussian mixture noise to further test misspecification of ϵ. The fourth setting considers the challenging case of (iv) a heteroscedastic Student-t noise with state-dependent degrees of freedom. In total we have $3 \times 4 \times 4 \times 6$ experiments (indexed by their dimensionality, noise setting, design heuristic, and metamodel type).

Besides the noise distribution, we fix all other metamodeling aspects. All schemes are initialized with $n_0 = 10d$ inputs drawn from an LHS design on $[0, 1]^d$ and use the SE kernel (3.4) for the covariance matrix $\boldsymbol{K}$. To analyze for the variability due to the initial design and the noise realizations, we perform 100 macroruns of each design/acquisition function combination. For each run, the same initial inputs are used across all GP metamodels and designs, but otherwise the initial $\mathbf{x}_{1:n_0}$ vary across runs.

Optimization of the Improvement Metric: We employed the MCU, ICU, tMSE and

Initial design	Latin hypercube sampling of size $n_0 = 10d$
Total budget n	$d = 1, n = 100; \quad d = 2, n = 150; \quad d = 6, n = 1000$
Test set size $M = \lvert\mathcal{D}\rvert$	$d = 1, M = 1000; \quad d = 2, M = 500; \quad d = 6, M = 1000$
Noise setting for $\epsilon(x)$	(i) t/small : $t_3(0, (0.1R_f)^2)$
	(ii) t/large : $t_3(0, R_f^2)$
	(iii) Gsn/mix: 50/50 mix of $\mathcal{N}(0, (0.5R_f)^2)$ and $\mathcal{N}(0, R_f^2)$
	(iv) t/hetero : $t_{6-4x^1}(0, (0.4(4x^1 + 1))^2)$

Table 4.4: Stochastic simulation setup for synthetic experiments. ($R_f \equiv \max_x f(x) - \min_x f(x)$)

cSUR criteria to maximize the improvement metric $\mathcal{I}$ and select the next input x_{n+1}. This maximization task is nontrivial in higher dimensions because $\mathcal{I}$ is frequently multimodal and can be flat around its local maxima. We use a genetic optimization approach as implemented in the `ga` function in MATLAB, with tolerance of 10^{-3} and 200 generations. This is a global, gradient-free optimizer that uses an evolutionary algorithm to explore the input space D.

Evaluation of Performance Metrics: Recall that evaluating the quality of $\partial\hat{S}$ is based on $\mathcal{ER}$ and $\mathcal{E}$ from (3.34) and (3.36) that require integration over D. In practice, these are computed based on a weighted sum over a finite $\mathcal{D}$, $\hat{\mathcal{E}} := \sum_{m=1}^{M} \Phi\big(\frac{-|\hat{f}(x_m)|}{s(x_m)}\big)\mu(x_m)$ for a space-filling sequence $\mathcal{D} \equiv x_{1:M} \in D$ of test points. In 1-D experiments $\mathcal{D}$ was an equispaced grid of size $M = 1000$. In higher dimensions, to avoid the use of a lot of test points that are required to ensure an accurate approximation, we adaptively pick $\mathcal{D}$ that targets the critical region close to the zero contour. To do so, we replace the integral with a weighted sum:

$$\mathcal{ER} \simeq \frac{p_c}{M_1} \sum_{x_{1:M_1} \in D_1} \mathbb{I}(\operatorname{sign} f(x_m) \neq \operatorname{sign} \hat{f}(x_m)) + \frac{(1-p_c)}{M_2} \sum_{x_{1:M_2} \in D_2} \mathbb{I}(\operatorname{sign} f(x_m) \neq \operatorname{sign} \hat{f}(x_m)), \tag{4.48}$$

where $M = M_1 + M_2$ and the test locations $x_{1:M_1}$ and $x_{1:M_2}$ are subsampled from a large space-filling (scrambled Sobol) sequence on D. The weight p_c determines the relative volume of D_1 and $D_2 = D \backslash D_1$, where on $D_1 = \{x : f(x) \simeq 0\}$ we are close to the zero contour. In the experiments below we use $M_1 = 0.8M, M_2 = 0.2M$, and $p_c = 0.4$, so that the density of test

points close to ∂S is double relative to those far from the zero contour. We employ the same strategy for speeding the evaluation of the empirical error $\mathcal{E}$.

Surrogate Inference: Values of hyperparameters $\boldsymbol{\vartheta}$ are crucial for good performance of GP metamodels. We estimate $\boldsymbol{\vartheta}$ using maximum likelihood. Except for TP, all models are fitted with the open source package `GPstuff` [104] in MATLAB. TPs are fitted with the `hetGP` [8] package in `R`. Auxiliary tests did not reveal any significant effects from using other available tools for plain GPs and t-GP, such as `GPML` [90].

In principle, the hyperparameters $\boldsymbol{\vartheta}$ change at every step of the sequential design, in other words, whenever $\mathcal{A}_n$ is augmented with (x_{n+1}, y_{n+1}). To save time however, we do not update $\boldsymbol{\vartheta}$ at each step. Instead, we first estimate the hyperparameters $\boldsymbol{\vartheta}$ based on the initial design $\mathcal{A}_{n_0}$ and then freeze them, updating their values only every few steps. Specifically, $\boldsymbol{\vartheta}$ is re-estimated at steps $n_0 + 1, n_0 + 2, n_0 + 4, n_0 + 8, n_0 + 16, \ldots$ (as the sample size becomes large, the inference of hyperparameters becomes more stable).

The lengthscales θ_i are the most significant for surrogate goodness of fit. A too-small lengthscale will make the estimated $\hat{f}$ look "wiggly" and might lead to overfitting, while θ_i too large will fail to capture an informative shape of the true f and hence S. Since our input domain is always $[0, 1]^d$, we restrict $\theta_i \in (0.1, 1)\ \forall i$ to be on the order of the length of the sample space D.

Computational Overhead: All the considered metamodels are computationally more demanding than the baseline Gaussian GP. For t-GP and Cl-GP, additional cost arises due to the Laplace approximation. TP necessitates estimation of the parameter ν and also the computation of β in (3.32). In the experiments considered, the respective computation times were roughly double to triple relative to the Gaussian GP. In terms of sequential design, MCU, tMSE, and cSUR have approximately equal overhead; ICU is significantly more expensive because it requires evaluating the sum in (4.19). Note that all heuristics include two expensive steps: optimization for x_{n+1} and computation of $\hat{f}^{(n)}$ and $s^{(n)}$ (and/or $\hat{s}^{(n+1)}$).

Overall timing of the schemes is complicated because of the combined effects of N (design budget), M (size of test set), and the use of different software (some schemes run in `R` and others in Matlab). Most important, the ultimate computation time is driven by the simulation cost of generating $Y(x)$-samples, which is trivial in the synthetic experiments but assumed to be large in the motivating context.

4.3.2 Comparison of GP Metamodels

Figure 4.5 shows the boxplots of the error rate $\mathcal{ER}$ of $\hat{S}^{(N)}$ at the final design ($N = 100$ in 1-D; $N = 150$ in 2-D; $N = 1000$ in 6-D). The plots are sorted by noise settings and design strategies, facilitating comparison between the discussed metamodels. In Table 4.5, we list the best metamodel and design combination in each case. Several high-level observations can be made. First, we observe the limitations of the baseline Gaussian GP metamodel, which cannot tolerate too much model misspecification. As the noise structure gets more complex, the classical GP surrogate begins to show increasing strain; in the last $t/hetero$ setup, it is both unstable (widely varying performance across runs) and inaccurate, with error rates upward of 30% on "bad" runs. In addition, according to results shown in Table 4.5, across all of the twelve cases, besides 1d example with $t/small$ noise, the Gaussian GP never performs as the best model. This result is not surprising but confirms that the noise distribution is key for the contour-finding task and illustrates the nonrobustness of the Gaussian observation model, due to which outliers strongly influence the inference.

Second, we document that the simple adjustment of using Student-t observations significantly mitigates the above issue. t-GP performs consistently and significantly better than Gaussian GP in essentially all settings. This result is true even when both models are misspecified (the Gsn/mix and $t/hetero$ cases). The performance of t-GP was still better (though not statistically significantly so) when we tested it in the setting of homoscedastic Gaussian

noise (not shown in the plots). The latter fact is not surprising—t-GP adaptively learns the degrees-of-freedom parameter ν and hence can "detect" Gaussian noise by setting ν to be large. Conversely, in heavy-tailed noise cases, the use of t samples will effectively ignore outliers [77] and thus produce more accurate predictions than working with a Gaussian observation assumption. We find that t-GP can handle complex noise structures and offers a good choice for all-around performance, making it a good default selection for applications. It brings smaller error rate $\mathcal{ER}$, more stable hyperparameter estimation, less contour bias, and tighter contour CI. Moreover t-GP is significantly better than all the other GPs in seven of the twelve setups, indicating that t-GP is essentially the best out of all GP metamodels in most cases.

Third, we also inspect the performance of the TP metamodel. As shown in Table 4.5, TP is the best in two cases out of the twelve, both of which are with the $t/small$ noise. In addition, TP has the smallest empirical error $\mathcal{E}$ (uncertainty) compared with the other metamodels in all cases except $t/hetero$ across 1-D and 2-D experiments. We note that TP works worst in $t/large$ and $t/hetero$ cases, having both large error rate $\mathcal{ER}$ and empirical error $\mathcal{E}$. Therefore, TP does not work well in cases with low signal-to-noise ratio or greatly misspecified noise.

Fourth, Cl-GP is also better than Gaussian GP in some cases with tMSE and MCU designs (except for the 6-D $t/hetero$ setup, where the error rate $\mathcal{ER}$ of MCU is not significantly different from that of ICU, although mean of ICU is slightly smaller). There is significant improvement for models with low signal-to-noise ratio; the only exception is for the low-noise setup where Cl-GP underperforms classical GP. This matches the intuition that employing classification "flattens" the signal by removing outliers. By considering only the sign of the response, the classification model largely disregards very large or highly negative observations, simplifying the noise at the cost of some information loss. The net effect is helpful when the noise is mis-specified or too strong so as to interfere with learning the mean response. The side effect is deleterious if the above gain is too little to outweigh the information loss, as apparently happens in the $t/small$ setup in 1-D and 6-D experiments. Of note, Cl-GP with MCU and MCl-GP

with MCU design have the smallest error rate among all GPs in two (Gsn/mix and $t/hetero$ in 1-D) out of 12 cases shown in Table 4.5. We also observe, however, that the stability of Cl-GP is highly dependent on the design: some designs create large across-run variations in performance. We hypothesize that this situation is linked to a more complex procedure for learning the hyperparameters of Cl-GP; therefore, designs that are not aggressive enough to explore the zero contour region (such as cSUR) can lead to difficulties in estimating ϑ. In particular, relative to t-GP, Cl-GP has more variable performance (i.e., larger sampling variance).

In terms of imposing monotonicity constraints, we observe two competing effects. On the one hand, as expected, monotonic GP surrogates generally reduce the error rate $\mathcal{ER}$ and the posterior uncertainty (hence $\mathcal{E}$) relative to the base surrogate. For example, in the 1-D example a monotonic surrogate will clearly assign the left edge $x \simeq 0$ to the negative level set N, greatly reducing $\bar{E}$ in that region compared to an unconstrained model. This effect is because the additional gradient constraint intrinsically lowers posterior uncertainty $s(x)$. On the other hand, monotonic GPs tend to exhibit greater bias in learning S. This phenomenon is notable in our experiments where the monotonic models have the worst bias across all metamodels. This occurs because the gradient constraints globally influence the shape of $\hat{f}$ and tend to make it flatter relative to f. As a result, observations far from the zero contour tend to systematically skew the latter's estimation.

4.3.3 Empirical Errors and Uncertainty Quantification

Figure 4.6 shows the empirical errors $\mathcal{E}$ that are supposed to proxy the true error rates $\mathcal{ER}$. Overall, we find that MCU tends to produce the largest $\mathcal{E}$, and ICU the smallest. These results are consistent with their design construction and local behavior: MCU heavily concentrates around $\partial\hat{S}$, which leads to little information collected about other regions, especially around the boundaries of sample space D and hence relatively large $\bar{E}(x)$ there, inflating $\mathcal{E}$. Conversely, the

objective function of ICU is precisely the myopic minimization of $\mathcal{E}_{n+1}$. The other two designs are intermediate versions in terms of minimizing $\mathcal{E}$. The tMSE heuristic tends to target the zero contour plus the edges of D, while cSUR tends to broadly target a "credible band" around $\partial\hat{S}$. Both approaches are better at reducing $\mathcal{E}$ compared with MCU but are not directly aimed at this. This logic is less consistent for the classification GPs, where tMSE often yields the lowest $\mathcal{E}$. This result echoes Section 4.3.2, namely, that classification GPs tend to perform better with MCU and tMSE designs in lower dimensional cases. TPs tend to have a much greater empirical error $\mathcal{E}$ when the noise is misspecified or in higher dimensional experiments, consistent to the conclusions obtained with the error rate $\mathcal{ER}$.

As a further visualization, Figure 4.7 shows the median error rate $\mathcal{ER}$ (3.34) and empirical error $\mathcal{E}$ in Eq. (3.36) as a function of step n in the 2-D Gsn/mix experiments. This illustrates the learning rates of different schemes as data is collected and offers a further comparison between the true $\mathcal{ER}$ and the self-reported $\mathcal{E}$ of the same scheme. We observe that some metamodels underperform for very low n, even if they eventually "catch up" after sufficiently large simulation budget. This is especially pronounced for the classification Cl-GP and MCl-GP metamodels, which yield very high $\mathcal{ER}^{(n)}$ (which is also much higher than the self-reported $\mathcal{E}$) for n small. We also note that TP, Cl-GP and MCl-GP all appear to enjoy faster reduction in $\mathcal{ER}^{(n)}$ compared with the baseline Gaussian GP, which we conjecture is due to better resistance against Y-outliers that distract plain GP's inference of S. Comparing the two rows of the figure, we note that discrepancies between $\mathcal{ER}$ and $\mathcal{E}$ tend to correlate with degraded performance, namely, the metamodel being unable to properly learn the response surface, and the poor uncertainty quantification leading to poor level set estimate. Moreover, the results suggest that the wedge in performance of different design criteria tends to persist; for example MCU and ICU frequently have not only the highest/lowest $\mathcal{E}^{(n)}$ but also the slowest/fastest rate of *reduction* in $\mathcal{E}^{(n)}$ as n grows. Consistent with results in Section 4.3.2, classification GPs with ICU criterion have both greater error rate $\mathcal{ER}$ and empirical error $\mathcal{E}$ in 2-D experiments.

4.3.4 Designs for Contour Finding

A key goal of our study is qualitative insights about experimental designs most appropriate for noisy level set estimation. Through identifying the best-performing heuristics we get an inkling regarding the structure of near-optimal designs for (2.2). In this section we illustrate the latter within a 2-D setup that can be conveniently visualized. Taking the $t/large$ experiment as an example, in Figure 4.8 we plot the fitted zero contour $\partial\hat{S}$ at $N = 150$ together with the chosen inputs $\mathbf{x}_{1:150}$ across the 6 metamodels and the 4 $\mathcal{I}$ heuristics. As might be expected, most of the designs are around the boundary of ∂S, which is the intrinsic way to minimize the error $\mathcal{ER}$. Nevertheless, we observe significant differences in designs produced by different $\mathcal{I}$'s. The MCU criterion places most of the samples close to the estimated zero contour $\partial\hat{S}$, reflecting its aggressive exploitation nature. For tMSE, the samples tend to cluster at several subregions of $\partial\hat{S}$ and on the edges of D. For cSUR, $\mathbf{x}_{1:n}$ cover a band along $\partial\hat{S}$, resembling the shape of MCU design but more dispersed. For ICU the design is much more exploratory, covering a large swath of D. All these findings echo the 1-D example in Figure 4.4.

One feature we observe is a so-called edge effect, that is, designs that focus on the edges of the input space. This effect arises due to the intrinsically high posterior uncertainty $s(x)$ for x close to ∂D. It features strongly in tMSE and cSUR (which have about 45% of the inputs along the edge) and to some extent in ICU (about 30% of inputs in this example). In contrast, MCU strongly discounts any region that is far from $\partial\hat{S}$. In the given 2-D experiment, we obtain some inputs directly on the boundary $\partial D = \{x^1 \in \{0, 1\} \cup \{x^2 \in \{0, 1\}\}$, that is, the maximizer of $\mathcal{I}_n(\cdot)$ lies exactly at its upper/lower bound (i.e. the constraint $x \in D$ is binding). A related phenomenon is the concentration of inputs in the top/left and bottom/right corners of D in the figure, which are associated with the highest uncertainty about the level set due to the confluence of the zero contour passing there and reduced spatial information from being on the edge of D.

Another noteworthy feature is *replication* of some inputs, that is, repeated selection of

the same $\mathbf{x}$ site. This does not occur for MCU, but happens for ICU, tMSE and cSUR that frequently (across macroruns) sample repeatedly at the vertices of D (indicated by the size of the corresponding marker in Figure 4.8). The replication is typically mild (we observe 145+ unique designs among a total of 150 x_n's). This finding echoes the importance for the metamodel to distinguish between signal and noise, which is a key distinction with the noise-free setting $\epsilon(x) \equiv 0$ [9].

Given this above discussion and the relative overhead associated with the different heuristics, we conclude that in lower dimensional problems, there is little benefit to using the more sophisticated ICU criterion, while for higher dimensional problems, ICU criterion is significantly better than the others. Beyond that, tMSE appears to be adequate and cheaper choices for 1D and 2D experiments. However, as the space becomes more complicated, we need more exploration over the input space and the explorative criteria like ICU start to shine.

The performance of designs differs when combined with different GP metamodels. Table 4.5 shows that there is not one overall “best" design for all metamodels across all cases. However, it does suggest some design/metamodel “combos" that work better than others, especially in the 1-D and 2-D experiments. The classification GPs seem to prefer more aggressive designs, such as MCU, while the regression GPs prefer more exploratory designs, such as ICU. In higher dimensions, ICU usually wins across all metamodels in accuracy; see the results of 6-D experiments in Table 4.5.

4.4 Application to Optimal Stopping Problems in Finance

In our next example we consider contour finding for determining the optimal exercise policy of a Bermudan financial derivative, as discussed in Section 2.2. The underlying simulator is based on a d-dimensional geometric Brownian motion $(\boldsymbol{X}_t)$ that represents asset prices and

follows the log-normal dynamics

$$\boldsymbol{X}_{t+\Delta t} = \boldsymbol{X}_t \exp\left((r - \frac{1}{2}\sigma^2)\Delta t + \boldsymbol{\Sigma}\Delta \boldsymbol{W}_t\right), \quad \Delta \boldsymbol{W}_t \sim \mathcal{N}(0, \Delta t \boldsymbol{I}), \tag{4.49}$$

where $\boldsymbol{I}$ is the $d \times d$ identity matrix. Let $h(t, x)$ be the option payoff from exercising when $\boldsymbol{X}_t = x$. Exercising is allowed every Δt time units, up to the option maturity T, so that we wish to determine $\{S_t : t \in \{\Delta t, 2\Delta t, \ldots, T - \Delta t\}\}$, which are the zero level sets of the timing function $x \mapsto T(t, x)$. During the backward dynamic programming, we iterate over $t = T, T - \Delta t, \ldots, 0$, and the simulator of $T(t, x)$ returns the difference between the pathwise payoff along a trajectory of $(\boldsymbol{X}_{t:T})$ that is based on the forward exercise strategy summarized by the forward-looking $\{\hat{S}_s, s > t\}$ and $h(t, x)$.

As discussed in [64], this setting implies a skewed, non-Gaussian, heteroscedastic distribution of the simulation noise and hence provides a challenging stochastic contour-finding problem. Note that in order to reflect the underlying distribution of $\boldsymbol{X}_t$ at time t (conditional on the given initial value $\boldsymbol{X}_0 = x_0$) the weighting measure $\mu(dx) = p_{X_t}(\cdot|x_0)$ is used. Thus, $\mu(\cdot)$ is log-normal based on (4.49) and is multiplied by the respective $\mathcal{I}_n$ criteria before selecting x_{n+1}. In line with the problem context, we no longer directly measure the accuracy of learning $\{S_t\}$ but instead focus on the ultimate output of RMC, which is the estimated option value in (2.6). The latter must itself be numerically evaluated via an out-of-sample Monte Carlo simulation that averages realized payoffs along a large database of M paths $x_{0:T}^{1:M}$:

$$\hat{V}(0, x_0) = \frac{1}{M}\sum_{m=1}^{M} h(\tau^m, x_{\tau^m}^{(m)}), \qquad \tau^m = \inf\{t : x_t^{(m)} \in \hat{S}_t\}. \tag{4.50}$$

Since our goal is to find the *best* exercise value, higher $\hat{V}$'s indicate a better approximation of $\{S_t\}$.

To allow a direct comparison, we set parameters matching the test cases in [64], considering

a 2-D and 3-D example. In both cases the volatility matrix $\boldsymbol{\Sigma} = \sigma \boldsymbol{I}$ in (4.49) is diagonal with constant terms; that is, the coordinates $\mathbf{X}^1_{1:n}, ..., \mathbf{X}^d_{1:n}$ are independently and identically distributed. As a first example, we consider a 2-D basket Put option with parameters $r = 0.06, \sigma = 0.2, \Delta t = 0.04, K = 40, T = 1$. The payoff is $h(t,x) = e^{-rt}(K - \frac{x^1+x^2}{2})_+$ with $K = 40$. Here it is known that stopping becomes optimal once both asset prices x^1 and x^2 become sufficiently low, so the level set S_t is always toward the bottom-left of D; see Fig 4.9. In contrast, stopping is definitely suboptimal when $h(t,x) = 0 \Leftrightarrow (x^1+x^2)/2 > K$. Consequently, the input sample space is taken to be $D = [25, 55] \times [25, 55] \cap \{x^1 + x^2 \leq 80\}$.

As a second example, we consider a 3-D max-Call with payoff $h(t,x) = e^{-rt}(\max(x^1, x^2, x^3) - K)_+$. The parameters are $r = 0.05, \delta = 0.1, \sigma = 0.2, X_0 = (90, 90, 90), K = 100, T = 3$ and $\Delta t = 1/3$. Since stopping is suboptimal when $h(t,x) = 0 \Leftrightarrow \max(x^1, x^2, x^3) < K$, the sample space is taken to be $D = [50, 150]^3 \times \{\max(x^1, x^2, x^3) > K\}$. In this case, stopping is optimal if *one* of the coordinates x^i is significantly higher than the other two, so S_t consists of three disconnected components. In this problem, there is no monotonicity, so we employ only the GP, t-GP, Cl-GP, and TP metamodels.

Because of the iterative construction of the simulator, the signal-to-noise ratio gets low for small t's. The variance $\tau^2(x)$ is also highly state-dependent, tending to be smaller for sites further from the zero-contour. To alleviate this misspecification and reduce metamodel overhead, we employ *batched* designs [64, 1], reusing $x \in D$ for r replications to collect observations $y^{(1)}(x), \ldots, y^{(r)}(x)$ from the corresponding simulator $Y(x)$. Then, we treat the mean of the r observations,

$$\bar{y}(x) = \frac{1}{r} \sum_{i=1}^{r} y^{(i)}(x), \tag{4.51}$$

as the response for input x and use $(x, \bar{y}(x))$ as a single design entry. The statistical properties of $\bar{y}$ are improved compared with the raw observations y: it is more consistent with the Gaussian

assumption thanks to the Central Limit Theorem (CLT), and its noise variance $\bar{\tau}^2(x) = \tau^2(x)/r$ is much smaller. Since the expense of sequential design of GP metamodels comes mainly from choosing the new input at each step, the reduction in budget $n = N/r$ by a factor of r significantly speeds their fitting and updating, with n for the number of unique inputs.More details about batching designs can be found in Chapter 5.

For the 2-D Put case study, we then test a total of three budget settings: (i) $r = 3, n = 80$ (low budget of $N = 240$ simulations); (ii) $r = 15, n = 80$ (high budget $N = 800$ with moderate replication); (iii) $r = 48, n = 25$ (high $N = 800$ with high replication). Comparing (ii) and (iii) shows the competing effects of having non-Gaussian noise (for lower r) and small design size (low n). The initial design size $n_0 = 10$. In this example, taking $n \gg 80$ gives only marginally better performance but significantly raises the computation time and hence is ruled out as impractical. Three setups are investigated for the 3-D example: $r = 3, n = 100$ (low-budget of $N = 300$), $r = 20, n = 100$ (moderate-budget of $N = 2000$) and $r = 20, n = 200$ (high budget $N = 4000$), both with $n_0 = 30$. In all examples, the results are based on 25 runs of each scheme and are evaluated through the resulting expected reward $\hat{V}(0, x_0)$ (4.50) on a fixed out-of-sample testing set of $M = 160,000$ paths of $\boldsymbol{X}_{0:T}$.

4.4.1 Results

Tables 4.6 and 4.7 compare the different designs and metamodels. To assess the sequential design gains, we also report the results from using a baseline nonadaptive LHS design on D. At low budget, we observe the dramatic gains of using adaptive designs for level set estimation, which allow us to obtain the same performance with an order-of-magnitude smaller simulation budget. The tMSE and cSUR criteria work best for the 2-D Put, while ICU is the best for the 3-D max-Call, indicating that the exploratory designs start to win out in more complex settings with higher d.

Regarding the metamodels, in the low-budget setups, the monotonic GP metamodel works best for the 2-D Put and t-GP for the 3-D max-Call. For the higher budget, which also coincides with higher $r \in \{10, 50\}$, the metamodel performance is similar, with t-GP slightly better than the other GP variants. In particular, once the SNR is high, classical Gaussian GP is effectively as good as any alternative. In both examples, TP and classification metamodels do not work well, possibly because of being more sensitive to the heteroscedastic aspect. We note that TP as well as the classification metamodels suffer from instability, so that lower $\hat{V}(0, x_0)$ is matched with a high sampling standard deviation. Another observation is that Cl-GP and MCl-GP perform badly with exploratory heuristic like ICU, especially with high budget.

Figure 4.9 shows the estimated exercise boundary $\partial\hat{S}_t$ with its 95% CI at $t = 0.4$ for the 2-D Put, for each of the five metamodels, each with the design yielding the highest payoff. We observe that all the best-performing designs look similar, placing about a dozen x_n's (some of which are from the initial design $x_{1:n_0}$) throughout D and the rest tightly along the zero contour. The results suggest that the criteria are largely interchangeable and that simpler $\mathcal{I}_n$ heuristics are able to reproduce the features of the more sophisticated or expensive ICU. The heuristics *do* differ in their uncertainty quantification; t-GP and GP generate tightest CI bands, while those of classification GPs and TP are too wide, indicating lack of confidence in the estimate. Of note, the regression GP metamodels (GP, t-GP and M-GP) also generate the lowest sampling variance for $\hat{V}(0, x_0)$.

Based on these results, our take-aways are threefold. First, similar to [64] we document significant gains from sequential design. Second, we find that while using ICU is helpful for more complicated settings with higher dimension d and larger budget, tMSE is the recommended DoE heuristic for lower dimensional cases, achieving excellent results with minimal overhead (in particular without requiring look-ahead variance). Third, we find that for applications with thousands of simulations, the Gaussian observation model is sufficient, since the underlying design needs to be replicated $r \gg 1$ in order to avoid excessively large $\mathbf{K}$-matrices. Therefore,

there is little need for more sophisticated metamodels, although useful gains can be realized from enforcing the monotonic structure, if available.

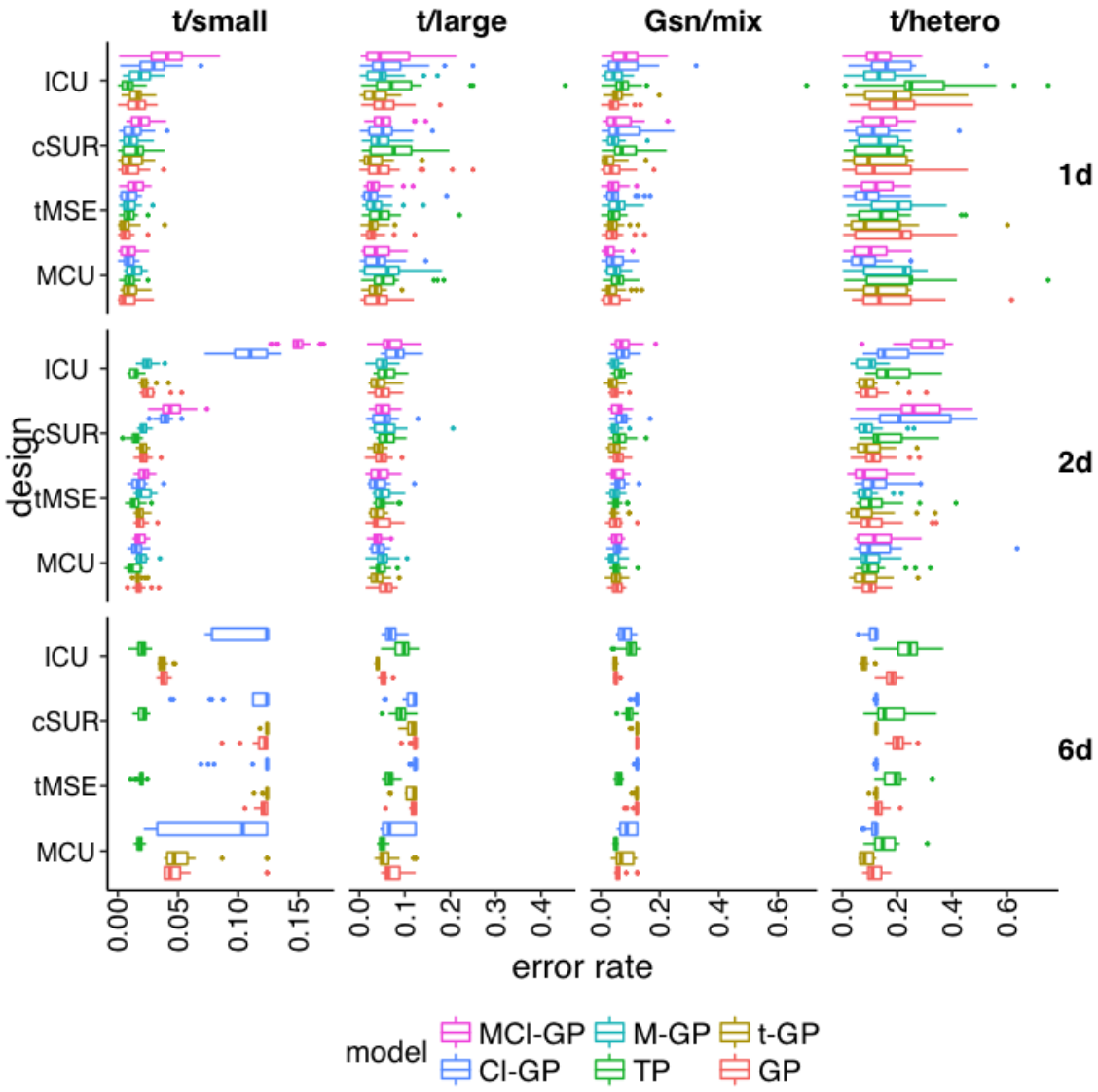


Figure 4.5: Boxplots of final error rate $\mathcal{ER}^{(\setminus)}$ from (3.34) across designs (rows) and noise setups (columns). Colors correspond to different GP metamodels. Note that x-axis limits are different across columns. Top row is for the 1-D experiment and design size $n = 100$; middle row: 2-D Branin-Hoo function with $n = 150$; bottom row: 6-D Hartman6 function with $n = 1000$.

Model	$t/small$		$t/large$		Gsn/mix		$t/hetero$	
				1-D *Quadratic*				
GP	*tMSE*	0.73% (0.60%)	tMSE	3.24% (2.79%)	MCU	3.87% (3.17%)	cSUR	15.68% (12.15%)
t-GP	tMSE	0.80% (0.93%)	*tMSE*	3.15% (1.83%)	cSUR	3.28% (3.74%)	cSUR	12.50% (9.05%)
TP	tMSE	1.02% (0.84%)	MCU	5.92% (5.63%)	tMSE	5.86% (4.99%)	cSUR	16.62% (11.35%)
Cl-GP	tMSE	0.87% (0.64%)	tMSE	3.39% (4.16%)	MCU	4.99% (3.77%)	*MCU*	8.83% (7.35%)
M-GP	tMSE	1.01% (0.72%)	tMSE	3.96% (3.31%)	ICU	4.73% (3.46%)	cSUR	13.45% (8.17%)
MCl-GP	MCU	0.99% (0.83%)	tMSE	3.82% (2.98%)	*MCU*	3.19% (2.74%)	MCU	10.89% (7.59%)
				2-D *Branin-Hoo*				
GP	MCU	1.78% (0.57%)	cSUR	4.75% (1.95%)	ICU	4.92% (1.86%)	MCU	10.36 % (3.94%)
t-GP	MCU	1.70% (0.29%)	*tMSE*	3.95% (1.47%)	*ICU*	4.10% (2.07%)	*tMSE*	9.00% (8.66%)
TP	*MCU*	1.27% (0.51%)	MCU	4.68% (1.54%)	tMSE	5.32% (1.75%)	MCU	11.90 % (7.40%)
Cl-GP	MCU	1.56% (0.51%)	MCU	4.27% (1.59%)	MCU	5.71% (1.85%)	tMSE	13.23% (7.74%)
M-GP	MCU	1.64% (0.33%)	ICU	4.92% (1.84%)	MCU	4.60% (2.26%)	tMSE	9.04% (4.60%)
MCl-GP	MCU	1.81% (0.47%)	MCU	4.26% (1.93%)	tMSE	5.39% (2.51%)	tMSE	10.72% (7.50%)
				6-D *Hartman6*				
GP	ICU	3.81% (0.34%)	ICU	5.33% (0.54%)	ICU	5.19% (0.70%)	MCU	11.67% (2.89%)
t-GP	ICU	3.75% (0.40%)	*ICU*	3.98% (0.47%)	*ICU*	4.86% (0.67%)	*ICU*	8.25% (1.60%)
TP	*MCU*	1.82% (0.28%)	MCU	5.05% (0.63%)	MCU	4.98% (0.59%)	MCU	15.56% (5.33%)
Cl-GP	MCU	7.99% (4.69%)	ICU	7.20% (0.66%)	ICU	8.31% (2.44%)	ICU	11.11% (2.20%)

Table 4.5: Mean (w/standard deviation) error rate $\mathcal{ER}$ and corresponding best-performing sequential design heuristic for the 1-D, 2-D, and 6-D synthetic case studies. Results are based on 100 macroreplications of each scheme.

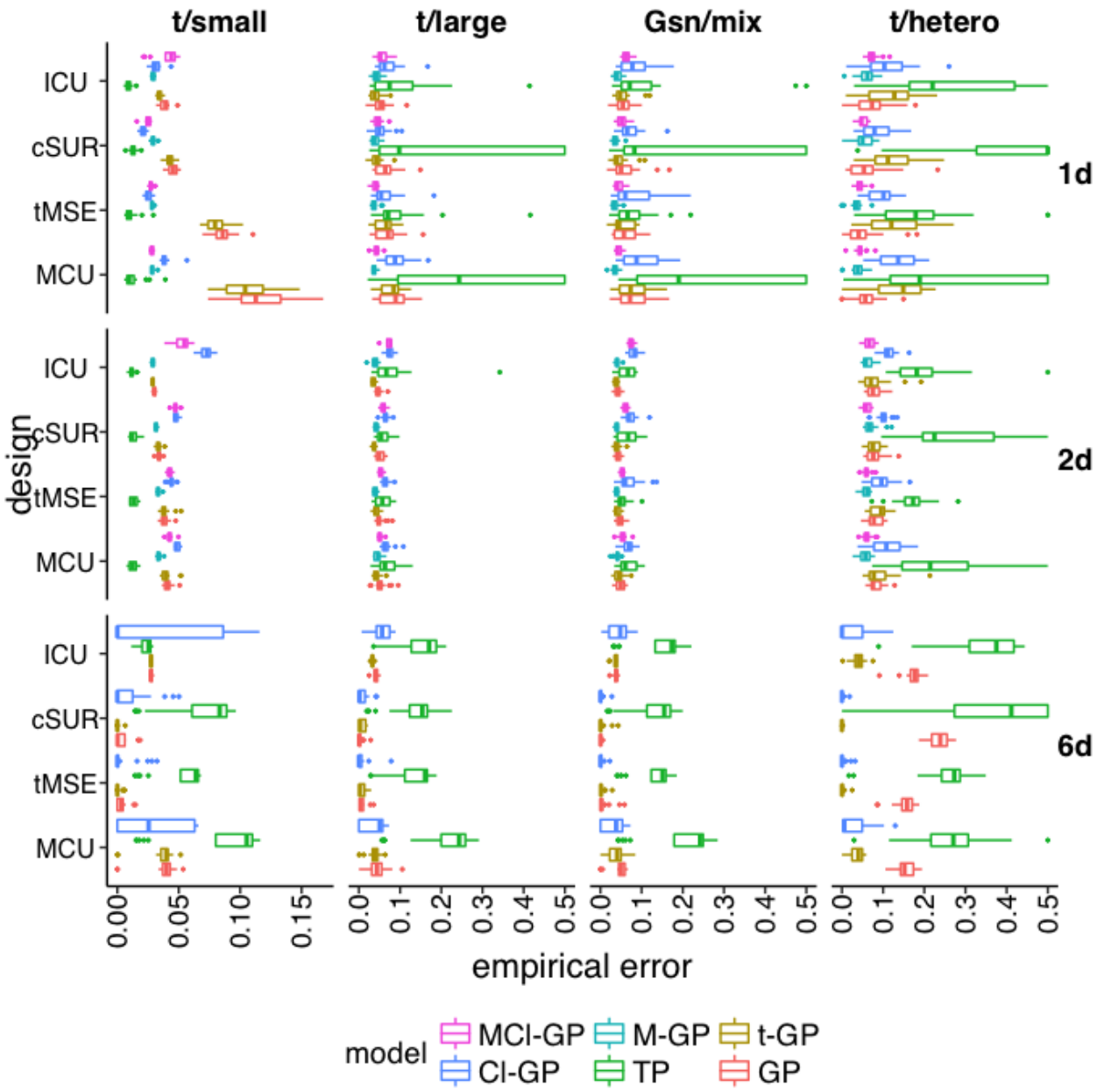


Figure 4.6: Empirical error $\mathcal{E}^{(n)}$ in Eq. (3.35) for GP, t-GP, TP, Cl- GP, and MCl-GP metamodels (colors), using MCU, tMSE, cSUR and ICU-based designs (sub rows) with $n = 100$ in 1-D , $n = 150$ in 2-D , and $n = 1000$ in the 6-D experiments (rows).

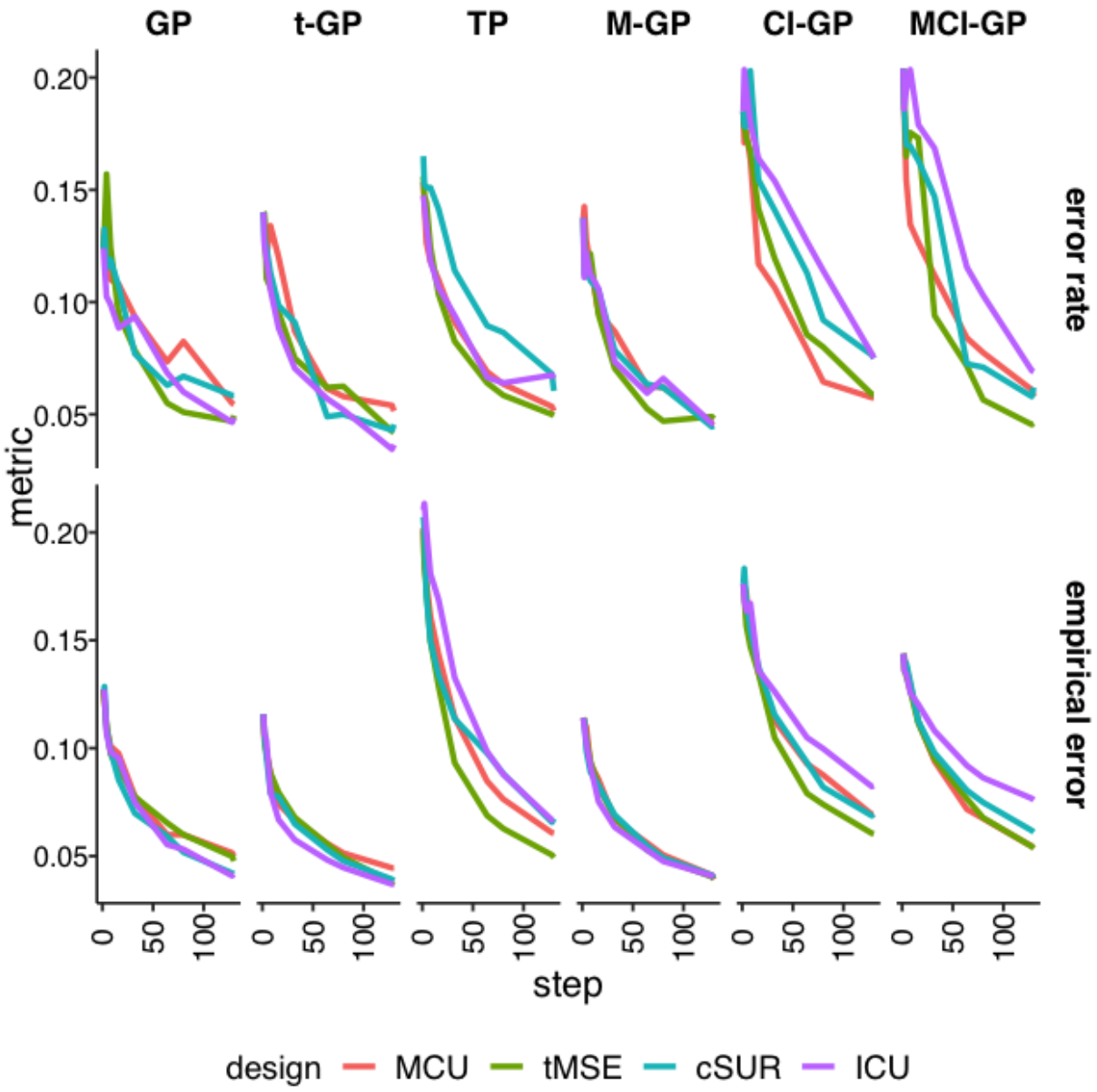


Figure 4.7: Error rate $\mathcal{ER}^{(\setminus)}$ (3.34) and surrogate-based uncertainty measure $\mathcal{E}^{(n)}$ (3.36) as a function of step n in the 2-D Gsn/mix setting. We compare six metamodels (columns) and four DoE's (colors). The y-axis limits differ across rows. We plot median results across 20 macroreplications of each scheme.

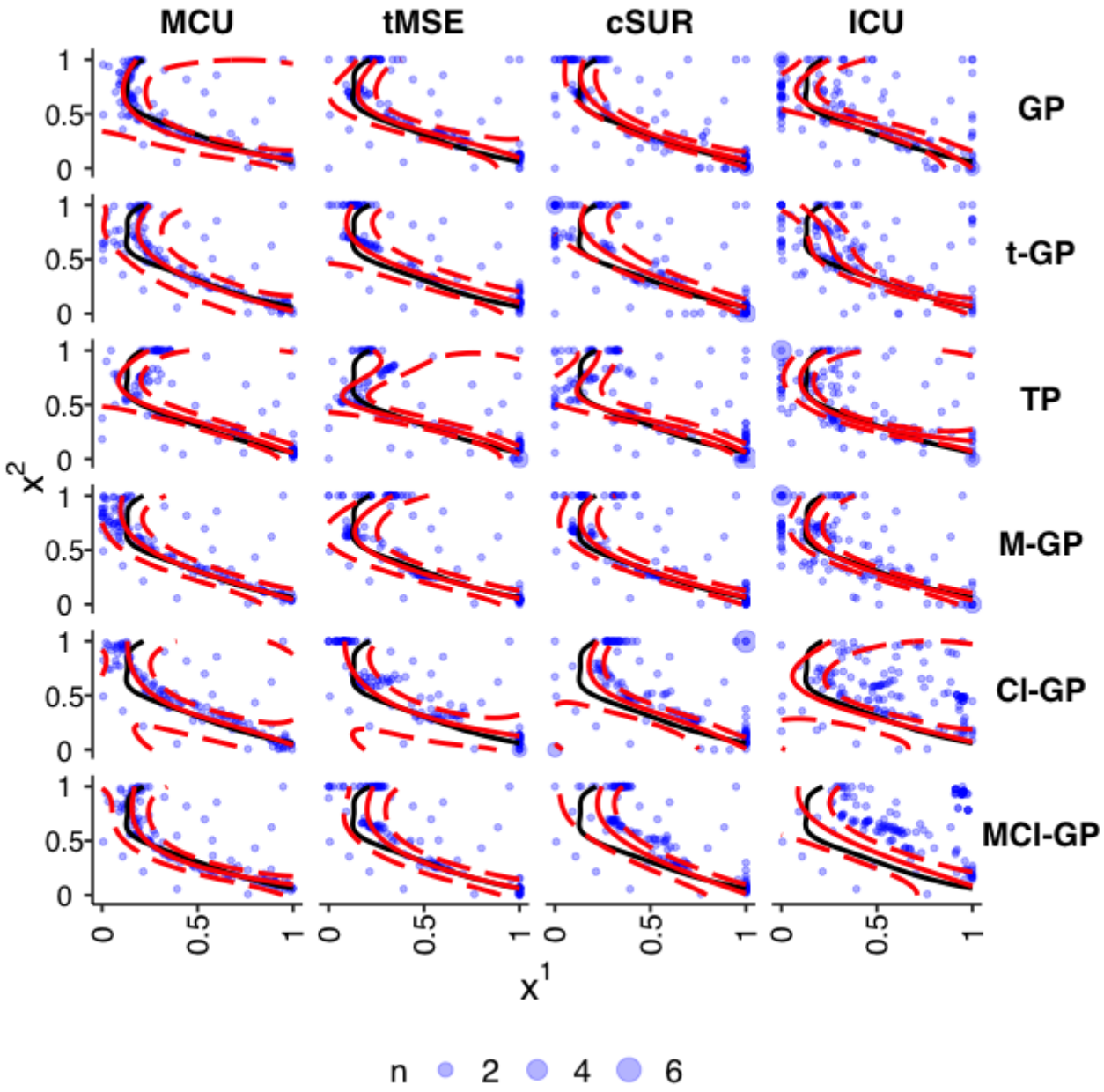


Figure 4.8: Estimates of the zero contour $\partial\hat{S}$ for the 2-D Branin-Hoo example with $t/large$ noise setting. We show $\partial\hat{S}^{(n)}$ (red solid line) at step $n = 150$, with its 95% credible band (red dashed lines), the true zero contour ∂S (black solid line) and the sampled inputs $\mathbf{x}_{1:n}$ (replicates indicated with correspondingly larger symbols). We compare across the six models (rows) and four DoE heuristics (columns).

	LHS	MCU	tMSE	cSUR	ICU
			$\mathbf{r = 3, n = 80}$		
GP	1.211(0.120)	1.425(0.008)	1.427(0.007)	1.431(0.009)	***1.431(0.007)***
t-GP	1.125(0.113)	1.409(0.013)	***1.417(0.008)***	1.409(0.010)	1.406(0.013)
TP	1.179 (0.133)	1.408 (0.022)	***1.414 (0.008)***	1.378 (0.044)	1.316 (0.037)
M-GP	1.403(0.014)	1.438(0.007)	1.440(0.006)	***1.442(0.009)***	1.433(0.005)
Cl-GP	1.111(0.121)	1.395(0.015)	***1.402 (0.013)***	1.393(0.013)	1.391(0.013)
MCl-GP	1.407(0.008)	1.429(0.010)	1.429(0.013)	***1.431(0.007)***	1.396(0.019)
			$\mathbf{r = 15, n = 80}$		
GP	1.425 (0.017)	1.448 (0.003)	***1.450 (0.002)***	1.450 (0.003)	1.449 (0.003)
t-GP	1.406 (0.033)	1.445 (0.003)	***1.447 (0.002)***	1.444 (0.005)	1.446 (0.004)
TP	1.414 (0.023)	***1.443 (0.003)***	1.443 (0.004)	1.441 (0.004)	1.430 (0.006)
M-GP	1.407 (0.008)	1.449 (0.003)	1.451 (0.002)	***1.454 (0.002)***	1.451 (0.003)
Cl-GP	1.353 (0.050)	***1.441 (0.004)***	1.440 (0.003)	1.435 (0.004)	1.436 (0.005)
MCl-GP	1.416 (0.010)	1.448 (0.004)	***1.449 (0.003)***	1.443 (0.003)	1.418 (0.008)
			$\mathbf{r = 48, n = 25}$		
GP	1.341 (0.068)	***1.450 (0.003)***	1.449 (0.003)	1.443 (0.004)	1.448 (0.003)
t-GP	1.336 (0.126)	1.449 (0.003)	***1.452 (0.003)***	1.442 (0.004)	1.449 (0.003)
TP	1.367 (0.063)	***1.433 (0.006)***	1.430 (0.011)	1.421 (0.039)	1.423 (0.023)
M-GP	1.415 (0.007)	***1.446 (0.002)***	1.444 (0.002)	1.445 (0.004)	1.442 (0.004)
Cl-GP	1.110 (0.144)	1.430 (0.010)	***1.434 (0.005)***	1.409 (0.008)	1.388 (0.016)
MCl-GP	1.423 (0.015)	1.446 (0.004)	***1.448 (0.003)***	1.413 (0.024)	1.414 (0.024)

Table 4.6: Performance of different designs and models on the 2-D Bermudan Put option in Section 4.4. Results are the mean (standard deviation) payoff of 25 runs of experiments evaluating on the same out-of-sample testing set of $M = 160000$ $\boldsymbol{X}_{0:T}$-paths at each run.

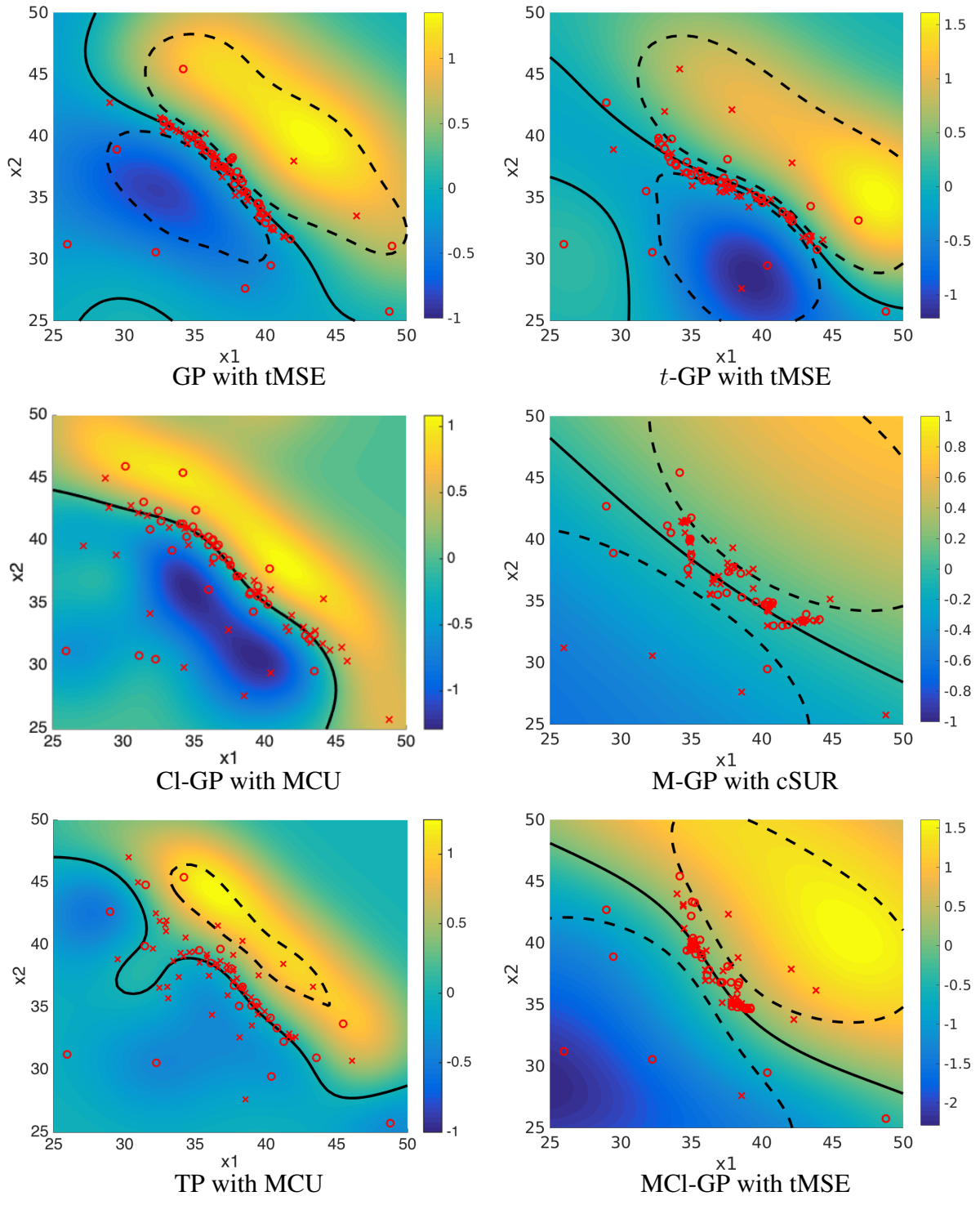


Figure 4.9: The estimated exercise boundary $\partial\hat{S}$ (solid line with 95% CI as dashed lines) at $t = 0.4$ for 2-D Bermudan Put from Section 4.4. Shading, which varies panel to panel, indicates the point estimate for the latent $\hat{f}(x)$ or $\hat{z}(x)$. We also show the design $(\mathbf{x}_{1:n}, \mathbf{y}_{1:n})$ with positive y_n's marked by $\times$ and negative y_n's by $\circ$. All schemes used $r = 15, n = 80$.

	LHS	MCU	tMSE	cSUR	ICU
			$\mathbf{r = 3, n = 100}$		
GP	10.036 (0.331)	10.725 (0.095)	10.773 (0.071)	10.711 (0.086)	***10.753 (0.072)***
t-GP	9.894 (0.447)	10.736 (0.088)	10.747 (0.087)	10.720 (0.104)	***10.782 (0.076)***
TP	9.169 (0.354)	10.101 (0.218)	9.872 (0.102)	8.867 (0.357)	***10.482 (0.156)***
Cl-GP	9.552 (0.567)	10.566 (0.084)	***10.657 (0.097)***	10.586 (0.099)	10.604 (0.119)
			$\mathbf{r = 20, n = 100}$		
GP	10.924 (0.076)	11.078 (0.029)	11.072 (0.028)	11.055 (0.032)	***11.101 (0.023)***
t-GP	10.923 (0.071)	11.061 (0.039)	11.055 (0.027)	11.044 (0.029)	***11.100 (0.027)***
TP	10.385 (0.178)	***10.815 (0.039)***	10.745 (0.045)	10.620 (0.087)	10.507 (0.087)
Cl-GP	10.761 (0.112)	***11.026 (0.032)***	10.991 (0.037)	10.901 (0.049)	10.937 (0.041)
			$\mathbf{r = 20, n = 200}$		
GP	11.105(0.036)	11.147(0.021)	11.119(0.022)	11.131(0.018)	***11.178(0.020)***
t-GP	11.090(0.034)	11.141(0.019)	11.126(0.020)	11.115(0.027)	***11.175(0.021)***
TP	10.585 (0.118)	***10.896 (0.030)***	10.811 (0.035)	10.764 (0.041)	10.638 (0.038)
Cl-GP	10.995(0.059)	11.109(0.025)	***11.056(0.040)***	10.985(0.027)	11.010(0.029)

Table 4.7: Performance of different designs and models on the 3-D Bermudan max-Call in Section 4.4. Results are the mean (w/standard deviation) payoff of 25 macroreplications evaluating on the same out-of-sample testing set of $M = 160000$ $\boldsymbol{X}_{0:T}$-paths at each run.

Chapter 5

Adaptive Batching for Gaussian Process Surrogates

5.1 Gaussian Processes with Batched Inputs

In Chapters 3 and 4, we benchmarked using GP-based metamodels on noisy level set estimation and developed four sequential design strategies with fast calculated look-ahead variance. The basic goal is to alleviate the misspecification of noise structure in model respective by replacing plain Gaussian Process with more robust metamodels. In Section 4.4, batching designs has shown to be significantly beneficial in cases with extremely low signal to noise ratio and large budget N. In this Chapter, we employ adaptive batching/replication to increase signal to noise ratio as well as decrease the metamodeling overhead, where at each input x, we simulate multiple observations and use their average as the final observation instead of generating only one observation.

To describe replicated inputs, let $\bar{x}_i$, $i = 1, ..., k$ denote the unique inputs, and $y_i^{(j)}$ be the j^{th} output of $r_i \geq 1$ replicates observed at $\bar{x}_i$. Let $\bar{\mathbf{y}}_{1:k} = \{\bar{y}_i, 1 \leq i \leq k\}$ store averages over replicates, $\bar{y}_i := \frac{1}{r_i} \sum_{j=1}^{r_i} y_i^{(j)}$. We still use the error rate $\mathcal{ER}$ defined in (3.34) to evaluate the

inference of level set S.

As discussed in Chapter 1, reconstructing S via a metamodel can be divided into two aspects: the construction of the response model $x \mapsto Y(x)$, and the development of the design of experiments (DoE) for efficiently selecting the inputs $\bar{x}_1, \bar{x}_2, \ldots$. To account for the second aspect, we use n to denote the rounds of sequential DoE, k_n to denote the number of unique inputs $\bar{x}$'s sampled by step n and $N_n = \sum_{i=1}^{k_n} r_i^{(n)}$ the respective number of simulator calls made. In Chapters 3 and 4, $k_n \equiv n$. The superscript on r_i allows the replicate counts to evolve over n as well, see Section 5.3. The metamodel training set by step n consists of $\mathcal{A}_n = \{(\bar{x}_i, r_i^{(n)}, \bar{y}_i), 1 \le i \le k_n\}$.

As defined in 3.1, the Gaussian process paradigm treats f as a random function whose posterior distribution is determined from its prior and the training set(s) $\mathcal{A}_n$. After incorporating the batch size r, the posterior mean $\hat{f}^{(n)}(x_*)$ (3.1) and covariance $v^{(n)}(x_*, x'_*)$ (3.2) at arbitrary inputs x_*, x'_* are given by

$$\hat{f}^{(n)}(x_*) = \mathbf{k}(x_*)[\mathbf{K} + \tau^2 \mathbf{R}^{(n)}]^{-1} \bar{\mathbf{y}}_{1:k_n}, \tag{5.1}$$

$$v^{(n)}(x_*, x'_*) = K(x_*, x'_*) - \mathbf{k}(x_*)[\mathbf{K} + \tau^2 \mathbf{R}^{(n)}]^{-1} \mathbf{k}(x'_*)^T, \tag{5.2}$$

with the $1 \times k_n$ vector $\mathbf{k}(x_*) = K(x_*, \bar{\mathbf{x}}_{1:k_n})$, the $k_n \times k_n$ matrix $\mathbf{K}$ given by $\mathbf{K}_{ij} = K(\bar{x}_i, \bar{x}_j)$, and the $k_n \times k_n$ diagonal matrix $\mathbf{R}^{(n)}$ given by $R_{ii}^{(n)} := \frac{1}{r_i^{(n)}}$. Comparing equations (3.1) and (5.1), and (3.2) and (3.2), we notice that the identity matrix $\mathbf{I}$ in (3.1) and (3.2) is changed into batch matrix $\mathbf{R}^{(n)}$. In fact, the identity matrix can be viewed as batch matrix with $r_i^{(n)} \equiv 1$, and thus the signal to noise ratio at each input is connected with its batch size. The posterior mean $\hat{f}^{(n)}(x_*)$ is still treated as a point estimate of $f(x_*)$, and the posterior standard deviation $s^{(n)}(x_*) := \sqrt{v^{(n)}(x_*, x_*)}$ as the uncertainty of this surrogate.

5.2 Adaptive Designs

5.2.1 Level Set Estimation

In this section we construct a sequential batched DoE to jointly select $(\bar{x}_{n+1}, r_{n+1})$. At each DoE round we pick a *new* input $\bar{x}_{n+1}$ and the associated replication amount r_{n+1}; thus by round n there are n unique inputs. In our first proposal, we formulate this task within a multi-fidelity framework, which is now widely used in Bayesian Optimization [53, 54, 55, 85]. Thanks to the LLN, we interpret r_n as *fidelity*: a small number of replicates is cheap but inaccurate; inputs with a large number of replicates are viewed as high-fidelity queries: expensive but accurate. Our interest is then to choose the fidelity level to query next, balancing the trade-off between accuracy and cost. As a second proposal, we relate replication to simulation and model fitting overhead costs, leading to maximization of the information gain $\mathcal{I}(x, r)$ per unit cost [57, 71].

Remark 4 *Another meaning of batched DoE refers to selecting multiple new inputs $\bar{x}_k$ in parallel, see [23]. In this article, batching always refers to using replicates; we add (at most) one new input at each DoE round.*

To begin, we recall two existing acquisition functions well suited to our needs. The first one is MCU criterion (4.14) which targets inputs with high response uncertainty (large $s^{(n)}(x)$), and close to the contour $\partial\hat{S}$ (small $|\hat{f}^{(n)}(x)|$). The cSUR criterion (4.17) focuses on quickly *reducing* the local empirical error $\bar{E}_n$ (3.37). These two criteria are selected due to their efficiency in computation, with one (MCU) based on the current data $\mathcal{A}_n$, and the other (cSUR) based on one-step ahead estimate on $\mathcal{A}_{n+1}$.

5.2.2 Fixed Batch Design

The basic strategy of choosing r_{n+1} is to predetermine a fixed value r_0 for it and keep it unchanged for all designs. We name it as Fixed Batch Design (FB). Ankenman et al. [1] showed

Algorithm 1 Multi-Level Batching (MLB)

Input: $\mathbf{r}_L$, η, k_0, r_0
$\mathcal{A}_{k_0} \leftarrow \{(\bar{x}_i, r_0, \bar{y}_i), 1 \le i \le k_0\}$, $(\hat{f}^{(k_0)}, s^{(k_0)}) \leftarrow f|\mathcal{A}_{k_0}$, $\gamma \leftarrow Ave(s^{(0)}(\bar{x}_{1:k_0}))$.
$N_{k_0} \leftarrow r_0 \times k_0$.
for $n = k_0, k_0 + 1, \ldots$ **do**
  $\bar{x}_{n+1} \leftarrow \arg\max_{x \in D} \mathcal{I}_n^{MCU}(x)$.
  $r_{n+1} \leftarrow \max\{r \in \mathbf{r}_L \,\big|\, s^{(n+1)}(\bar{x}_{n+1}, r) \ge \gamma\}$.
  while r_{n+1} does not exist **do**
    $\gamma \leftarrow \eta \times \gamma$.
    $r_{n+1} \leftarrow \max\{r \in \mathbf{r}_L \,\big|\, s^{(n+1)}(\bar{x}_{n+1}, r) \ge \gamma\}$.
  end while
  $\bar{y}_{n+1} \leftarrow \frac{1}{r_{n+1}} \sum_{j=1}^{r_{n+1}} y^{(j)}$.
  Update $\mathcal{A}_{n+1} \leftarrow \mathcal{A}_n \cup \{(\bar{x}_{n+1}, r_{n+1}, \bar{y}_{n+1})\}$.
  Obtain $(\hat{f}^{(n+1)}, s^{(n+1)}) \leftarrow f|\mathcal{A}_{n+1}$.
  $N_{n+1} \leftarrow N_n + r_{n+1}$.
end for

that selecting r_0 is a sensitive issue in stochastic surface metamodeling. In our research, we combine FB with acquisition function of MCU. A small value of replicates r_{n+1} poses a danger of lacking exploitation in the targeted region close to zero contour. On the other hand, with a large value of r_{n+1} we might be left with too few unique designs, resulting in an inadequate exploration of the entire space. Therefore, we would like to find an adaptive way to determine the value of r_{n+1} together with the input x_{n+1}, so that the value adjusts to the input as well as the design size n.

5.2.3 Multi-Level Batching

To improve upon FB, we select r_{n+1} from a discrete set $\mathbf{r}_L := \{r^1, \ldots, r^L\}$, interpreted as representing L different *sampling fidelities*. Query at x on the ℓ-th level implies using r^ℓ replicates to generate observations $y^{(j)}, j = 1, \ldots, r^\ell$ yielding the average $\bar{y}$. The cost of the ℓ-th fidelity is proportional to r^ℓ.

Kandasamy et al. [53] investigated multi-fidelity GP metamodels, with the idea of using low/cheap fidelities to explore and then high/expensive fidelities to exploit the desired contour.

Using this strategy, we propose the MLB Algorithm 1 which first chooses the next input $\bar{x}_{n+1}$ and afterwards the associated number of replicates r_{n+1}. Specifically, we propose to determine $\bar{x}_{n+1}$ via the MCU criterion $\mathcal{I}_n^{MCU}$ (4.14) and then choose r_{n+1} based on the look-ahead standard deviation $s^{(n+1)}(\bar{x}_{n+1}, \cdot)$ in (4.25). If $s^{(n+1)}(\bar{x}_{n+1}, \cdot)$ is large, we want to favor exploration; conversely well-explored regions will have low $s^{(n+1)}(\bar{x}_{n+1}, \cdot)$ and should lead to larger r_{n+1}. The choice of r_{n+1} is based on a threshold $\gamma = \gamma_n$, namely choosing the highest fidelity r^ℓ which has look-ahead $s^{(n+1)}(\bar{x}_{n+1}, r^\ell)$ still greater than γ, see Algorithm 1. If $s^{(n+1)}(\bar{x}_{n+1}, r^L) > \gamma$ then we use the highest fidelity level; if $s^{(n+1)}(\bar{x}_{n+1}, r^1) < \gamma$, then we lower the threshold by multiplying γ by a reduction factor $\eta < 1$, and try to pick r_{n+1} again, cf. [53]. Note that unlike other acquisition functions, including cSUR, MCU is based solely on information in $\mathcal{A}_n$ and hence allows to decouple the selection of $\bar{x}_{n+1}$ from that of r_{n+1}.

5.2.4 Ratchet Batching

By construction, the MLB Algorithm 1 will step back and forth between different replication levels r^ℓ. Since intuitively the design should concentrate as n grows, we expect r_n to grow over time. This suggests a variant of MLB that restricts $n \mapsto r_n$ to be monotonically non-decreasing and reduces picking r_{n+1} among just two fidelity levels (compared to L levels in MLB). The resulting Ratchet Batching (RB) scheme is summarized in Algorithm 2, which replaces the $\max$ in Algorithm 1 with an if-else statement for either keeping $r_{n+1} = r_n = r^\ell$ or incrementing to a higher replication level if $s^{(n+1)}(\bar{x}_{n+1}, r_n^\ell)$ is above γ. As in MLB, the latter threshold γ is progressively lowered whenever $s^{(n+1)}(\bar{x}_{n+1}, r_n^\ell) < \gamma$. For RB, the variance decrement factor η for γ should be chosen to be larger, to avoid excessive ratcheting up in r_n. If η is not large enough, there is a risk to skip levels in $\boldsymbol{r}_L$ and to end up with excessive replication relative to number of simulation calls, leading to insufficient exploration.

Algorithm 2 Ratchet Batching (RB)

Input: $\mathbf{r}_L$, η, k_0, r_0
$\mathcal{A}_{k_0} \leftarrow \{(\bar{x}_i, r_0, \bar{y}_i), 1 \le i \le k_0\}$, $(\hat{f}^{(k_0)}, s^{(k_0)}) \leftarrow f|\mathcal{A}_{k_0}$, $\gamma \leftarrow s^{(k_0)}$.
$N_{k_0} \leftarrow r_0 \times k_0$.
for $n = k_0, k_0 + 1, \ldots$ **do**
  $\bar{x}_{n+1} \leftarrow \arg\max_{x \in D} \mathcal{I}_n^{MCU}(x)$.
  while $\gamma > s^{(n+1)}(\bar{x}_{n+1}, r_n)$ **do**
    $\gamma \leftarrow \eta \times \gamma$.
  end while
  if $\gamma > s^{(n+1)}(\bar{x}_{n+1}, r_n^{\ell+1})$ **then**
    $r_{n+1} \leftarrow r_n$.
  else
    $r_{n+1} \leftarrow r_n^{\ell+1}$.
  end if{ $r_n^{\ell+1}$ represents the next level above r_n in $\mathbf{r}_L$}
  $\bar{y}_{n+1} \leftarrow \frac{1}{r_{n+1}} \sum_{j=1}^{r_{n+1}} y^{(j)}$.
  Update $\mathcal{A}_{n+1} \leftarrow \mathcal{A}_n \cup \{(\bar{x}_{n+1}, r_{n+1}, \bar{y}_{n+1})\}$.
  Obtain $(\hat{f}^{(n+1)}, s^{(n+1)}) \leftarrow f|\mathcal{A}_{n+1}$.
  $N_{n+1} \leftarrow N_n + r_{n+1}$.
end for

5.2.5 Adaptively Batched Stepwise Uncertainty Reduction

The FB, MLB and RB schemes all pick $\bar{x}_{n+1}$ first and then r_{n+1}. We next propose a procedure to pick both through a joint optimization criterion. The main idea is to tie the choice of r_{n+1} to *cost*, namely to maximize the ratio of the information gain and the cost of generating r outputs, plus the optimization overhead. The inclusion of the overhead in $\mathcal{I}_n$ comes from Klein et al. [57] and McLeod et al. [71], where the authors treated the total cost as the sum of query cost T_{sim} and the GP metamodeling overhead c_{over}. This was then extended by Swersky et al. [100] to multi-fidelity Bayesian Optimization. Stroh [99] discussed estimating a probability of exceeding a threshold in a multi-fidelity stochastic simulator, where the input $\bar{x}_{n+1}$ and the fidelity are estimated in a sequential way. We develop an analogue for level-set estimation via a

cSUR-based acquisition function

$$\mathcal{I}_n^{ABSUR}(x,r) := \frac{\mathcal{I}_n^{cSUR}(x,r)}{c(r)+c_{over}(n)}, \tag{5.3}$$

where $c_{over}(n)$ is the overhead and $c(r) = r \cdot T_{sim}$ is the cost of r evaluations, linear in r. Combining (5.3) and (4.26), we obtain

$$\mathcal{I}_n^{ABSUR}(x,r) := \frac{\Phi\left(-\frac{|\hat{f}^{(n)}(x)|}{s^{(n)}(x)}\right) - \Phi\left(-\frac{|\hat{f}^{(n)}(x)|}{s^{(n)}(x)}\frac{\sqrt{rs^{(n)}(x)^2+\tau^2}}{\tau}\right)}{r \cdot T_{sim} + c_{over}(n)}. \tag{5.4}$$

The resulting ABSUR Algorithm 3 myopically maximizes $\mathcal{I}^{ABSUR}$ over $x \in D$ and $r \in \mathcal{R} = [\underline{r}, \bar{r}]$. Intuitively, the location of $\bar{x}_{n+1}$ is similar to the cSUR DoE and the value of r_{n+1} is controlled by $s^{(n)}(x)^2$ and $c_{over}(n)$; more replication results when $s^{(n)}(x)^2$ is small or $c_{over}(n)$ is large.

Algorithm 3 Adaptive Batched SUR (ABSUR)

Input: $\mathcal{R} = [\underline{r}, \bar{r}]$, k_0, r_0, $n \mapsto c_{over}(n)$, T_{sim}
$\mathcal{A}_{k_0} \leftarrow \{(\bar{x}_i, r_0, \bar{y}_i), 1 \le i \le k_0\}$, $(\hat{f}^{(k_0)}, s^{(k_0)}) \leftarrow f|\mathcal{A}_{k_0}$
$N_{k_0} \leftarrow r_0 \times k_0$
for $n = k_0, k_0+1, \ldots$ **do**
  $(\bar{x}_{n+1}, r_{n+1}) \leftarrow \arg\sup_{x \in D, r \in \mathcal{L}} \mathcal{I}_n^{ABSUR}(x,r)$.
  $\bar{y}_{n+1} \leftarrow \frac{1}{r_{n+1}} \sum_{j=1}^{r_{n+1}} y^{(j)}$.
  Update $\mathcal{A}_{n+1} \leftarrow \mathcal{A}_n \cup \{(\bar{x}_{n+1}, r_{n+1}, \bar{y}_{n+1})\}$.
  Obtain $(\hat{f}^{(n+1)}, s^{(n+1)}) \leftarrow f|\mathcal{A}_{n+1}$.
  $N_{n+1} \leftarrow N_n + r_{n+1}$.
end for

There are four hyperparameters in ABSUR: the simulation cost T_{sim}, the overhead cost function $c_{over}(n)$ and the lower/upper bounds of replication $[\underline{r}, \bar{r}]$. For $c_{over}(n)$ we follow the recipe in McLeod et al. [71], modeling it as a quadratic function of n to reflect the prediction

complexity of GPs:

$$c_{over}(n;\boldsymbol{\theta}) = \theta_0 + \theta_1 n + \theta_2 n^2, \tag{5.5}$$

where $\boldsymbol{\theta}$ are fitted empirically. Alternatively Klein et al. [57] kept $c_{over}(n)$ as a constant. The value of T_{sim} represents the cost of obtaining each observation. If simulations are cheap, we would like to replicate more, and indeed lower T_{sim} leads to smaller designs. This feature implies that T_{sim} should be larger in higher-dimensional settings, since more unique inputs are needed to explore a larger input space.

5.3 Adaptive Design with Stepwise Allocation

The four strategies (FB, MLB, RB and ABSUR) discussed in Section 5.2 visit each input site $\bar{x}_{n+1}$ only once. Consequently, the respective replicate count r_{n+1} is determined at step $n+1$ and then remains the same throughout the latter steps. As an alternative, one can sequentially *allocate* new simulations across existing designs, thereby gradually growing $r_i^{(n)}$. Namely, the algorithm identifies existing "informative" inputs and augments their replicate counts, without changing the number of unique inputs k_n across the sequential design rounds n. In our context, we pair this augmentation with the option of expanding the design set itself. This ADSA approach resembles *Stepwise Approximate Optimal Design (SAO)*, an IMSE-based sequential design strategy proposed by Chen and Zhou [22] for surface metamodeling.

At each step n we are given a budget of $\Delta r^{(n)}$ additional simulations, and the main decision is to determine whether we should choose a new input $\bar{x}_{k_n+1}$ that then receives all $\Delta r^{(n)}$ replicates, or we should allocate $\Delta r^{(n)}$ simulator calls across the existing inputs $\bar{\mathbf{x}}_{1:k_n}$. In the latter case, we aim to minimize the global look-ahead integrated contour uncertainty $\mathcal{L}^{(n+1)}$

where the metric $\mathcal{L}^{(n)}$ is defined by

$$\mathcal{L}^{(n)} := \sum_{j=1}^{M} \omega_j^{(n)} \hat{f}^{(n)}(x_{j,*}) = (\boldsymbol{\omega}^{(n)})^T \mathbf{f}_*^{(n)} \simeq \int_D \omega^{(n)}(x) \hat{f}^{(n)}(x) dx, \tag{5.6}$$

where $\mathbf{x}_* = x_{1,*}, \ldots, x_{M,*}$ is a test set of size M (constructed using Latin Hybercube Sampling), $\mathbf{f}_*^{(n)} \equiv \hat{f}(\mathbf{x}_*)$ is the predicted response at $\mathbf{x}_*$, and

$$\omega_j^{(n)} \equiv \omega(x_{j,*})\mu(x_{j,*}) := \Phi(-\hat{f}^{(n)}(x_{j,*})/s^{(n)}(x_{j,*}))\mu(x_{j,*})$$

are the weights that target the level-set region of interest (compare to the targeted integrated mean square error (tIMSE) criterion proposed in Picheny et al. [83]).

For allocation purposes, we approximate the look-ahead $\mathcal{L}^{(n+1)}$ as a linear combination of the M predictions $\hat{f}^{(n+1)}(x_{j,*})$ with *fixed* importance weights $\boldsymbol{\omega}^{(n)}$, whereby our goal is to minimize the variance of $(\boldsymbol{\omega}^{(n)})^T \mathbf{f}^{(n+1)}$ conditional on the extra allocations $\Delta r_i^{(n)}$ at each input $\bar{x}_i$. Since the covariance matrix of $\mathbf{f}_*^{(n+1)}$ given replication counts $\mathbf{R}^{(n+1)}$ is

$$\mathbf{C}^{(n+1)} = \mathbf{k}(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*, \bar{\mathbf{x}}_{1:k_n})(\boldsymbol{K} + \tau^2 \mathbf{R}^{(n+1)})^{-1} \mathbf{k}(\mathbf{x}_*, \bar{\mathbf{x}}_{1:k_n})^T \tag{5.7}$$

the objective becomes the quadratic program

$$\mathcal{L}^{(n+1)}((\Delta r_i)_{i=1}^{k_n}) = (\boldsymbol{\omega}^{(n)})^T \mathbf{C}^{(n+1)} \boldsymbol{\omega}^{(n)} \mapsto \min! \tag{5.8}$$

under the constraint $\sum_i \Delta r_i^{(n)} = \Delta r^{(n)}$.

Define the $k_n \times k_n$ matrix $\boldsymbol{\Sigma}^{(n)} = \mathbf{K} + \tau^2 \mathbf{R}^{(n)}$ and the $M \times k_n$ matrix $\mathbf{K}_* := \mathbf{K}(\mathbf{x}_*, \bar{\mathbf{x}}_{1:k_n})$. The next proposition, proven in Section 5.3.1, explains how to pick $\Delta r_i^{(n)}$'s to minimize (5.8).

Proposition 5.3.1 *Let $\Delta \mathbf{R}^{(n)} := \mathbf{R}^{(n)} - \mathbf{R}^{(n+1)}$ be a $k_n \times k_n$ diagonal matrix with ele-*

ments $\Delta\mathbf{R}_{ii}^{(n)} = \frac{\Delta r_i^{(n)}}{(r_i^{(n)}+\Delta r_i^{(n)})r_i^{(n)}} = [r_i^{(n)}]^{-1} - (r_i^{(n)} + \Delta r_i^{(n)})^{-1}$, $i = 1, \ldots, k_n$. *Assume* $\max_{i=1,\ldots,k_n} \Delta\mathbf{R}_{ii}^{(n)} \ll 1$. *The optimal allocation rule that minimizes* (5.8) *is to assign* $\Delta r_i^{(n)}$ *to each* $\bar{x}_i$ *such that*

$$r_i^{(n)} + \Delta r_i^{(n)} \propto \mathbf{U}_i^{(n)}, \tag{5.9}$$

where

$$\mathbf{U}^{(n)} = (\mathbf{\Sigma}^{(n)})^{-1}\mathbf{K}_*^T\boldsymbol{\omega}^{(n)}. \tag{5.10}$$

Algorithm 4 Adaptive Design with Stepwise Allocation (ADSA)

Input: $\bar{\mathbf{x}}_*$, $\bar{\mathbf{x}}_{1:k_0}$, k_0, r_0, $c_{batch} = \frac{20}{d}$
$\mathcal{A}_{k_0} \leftarrow \{(\bar{x}_i, r_0, \bar{y}_i), i = 1, ..., k_0\}$. $(\hat{f}^{(k_0)}, s^{(k_0)}) \leftarrow f|\mathcal{A}_{k_0}$, $N_0 \leftarrow r_0 \times k_0$.
for $n = k_0, k_0 + 1, \ldots$ **do**
  $\Delta r^{(n)} \leftarrow c_{batch}\sqrt{k_0 + n}$.
  Calculate allocations $\Delta r_i^{(n)}, 1 \le i \le k_n$ with Algorithm 5.
  $\bar{x}_{k_n+1} \leftarrow \arg\max_{x\in D} \mathcal{I}_n^{MCU}(x, \Delta r^{(n)})$.
  Calculate $\mathcal{I}_{ADSA}^{(n)-all}, \mathcal{I}_{ADSA}^{(n)-new}$ in (5.13) and (5.11).
  Case 1:
    New $\bar{y}_{k_n+1} \leftarrow \frac{1}{\Delta r^{(n)}} \sum_{j=1}^{\Delta r^{(n)}} y^j(\bar{x}_{k_n+1})$.
    Update $\mathcal{A}_{n+1} \leftarrow \mathcal{A}_n \cup \{(\bar{x}_{k_n+1}, \Delta r^{(n)}, \bar{y}_{k_n+1})\}$.
    $N_{n+1} \leftarrow N_n + \sum_i \Delta r_i^{(n)}$ (May not be exactly $\Delta r^{(n)}$).
    $k_{n+1} \leftarrow k_n + 1$.
  Case 2:
    For $i = 1, ..., k_n$, update $\bar{y}_i \leftarrow \frac{\bar{y}_i \times r_i^{(n)} + \sum_{j=1}^{\Delta r_i^{(n)}} y^j(\bar{x}_i)}{r_i^{(n)} + \Delta r_i^{(n)}}$, $r_i^{(n+1)} \leftarrow r_i^{(n)} + \Delta r_i^{(n)}$
    Update $\mathcal{A}_{n+1} \leftarrow \{(\bar{x}_i, r_i^{(n+1)}, \bar{y}_i)\}_{i=1,\ldots,k_n}$.
    $N_{n+1} \leftarrow N_n + \sum_{i=1}^{k_n} \Delta r_i^{(n)}$
    $k_{n+1} \leftarrow k_n$
  Obtain $(\hat{f}^{(n+1)}, s^{(n+1)}) \leftarrow f|\mathcal{A}_{n+1}$.
  ADSA: **Do** Case **1 if** $\mathcal{I}_{ADSA}^{(n)-all} > \mathcal{I}_{ADSA}^{(n)-new}$, **otherwise do** Case **2**
  {FDSA variant:} **Do** Case **2**.
  {DDSA variant:} **Do** Case **1** if n is odd, Case **2** if n is even.
end for

After obtaining the allocations $\Delta\mathbf{r}^{(n)}_{1,\ldots,k_n}$, we compute the resulting look-ahead tIMSE metric:

$$\mathcal{I}^{(n)-all}_{ADSA} := \sum_{j=1}^{M} \tilde{s}^{(n+1)}(x_{j,*})^2 \omega_j^{(n)}, \tag{5.11}$$

where the look-ahead variance $\tilde{s}^{(n+1)}(\cdot)^2$ is based on the new replicate counts $r_i^{(n+1)} = r_i^{(n)} + \Delta r_i^{(n)}, i = 1, \ldots, k_n$, see proof in [25], [44] and [67]:

$$\tilde{s}^{(n+1)}(\mathbf{x}_*)^2 = s^{(n)}(\mathbf{x}_*)^2 - \mathbf{k}_*(\mathbf{\Sigma}^{(n)})^{-1}\Delta\mathbf{R}^{(n)}(\mathbf{\Sigma}^{(n)})^{-1}\mathbf{k}_*^T. \tag{5.12}$$

The alternative to allocating over existing $\bar{\mathbf{x}}_{1:k_n}$ is to pick a new input x_{k_n+1} and assign it $\Delta r^{(n)}$ simulations. To do so, we use the MCU criterion to make it consistent with FB, MLB and RB. (Other acquisition functions can also be used and experiments suggest that the algorithm is not sensitive to this choice.) Then we evaluate the resulting $\mathcal{I}^{(n)-new}_{ADSA}$:

$$\mathcal{I}^{(n)-new}_{ADSA} := \sum_{j=1}^{M} s^{(n+1)}(x_{j,*}, \Delta r^{(n)})^2 \omega_j^{(n)}, \tag{5.13}$$

$$s^{(n+1)}(x_{j,*}, \Delta r^{(n)})^2 = s^{(n)}(x_{j,*})^2 - \frac{v^{(n)}(x_{j,*}, \bar{x}_{k_n+1})^2}{\frac{\tau^2}{\Delta r^{(n)}} + s^{(n)}(\bar{x}_{k_n+1})^2},$$

where the sums are used as approximations of the underlying integrals over $x \in D$. Finally, we compare $\mathcal{I}^{(n)-new}_{ADSA}$ and $\mathcal{I}^{(n)-all}_{ADSA}$ to determine whether to sample at the new $\bar{x}_{k_n+1}$ or to allocate to existing $\mathbf{x}_{1:k_n}$, picking the maximum of the two tIMSE metrics.

For FB, MLB, RB and ABSUR, as we select one new input at each step, we have $k_n = n$. However, for ADSA we either select a new input or re-allocate, so that the resulting design size satisfies $k_n < n$. Thus, relative to the earlier schemes, in ADSA the size of $\mathcal{A}_n$ and the number of DoE rounds n are no longer deterministically linked and the number of unique inputs is endogenous to the particular algorithm run.

A major goal of all our schemes is for k_n to grow sub-linearly in n, i.e. fewer and fewer

new inputs are added as more simulations are run. In ADSA, this translates into endogenously preferring re-allocation over adding inputs as n. The user can further preference this situation by making the batches $\Delta r^{(n)}$ also grow in n. In practice we observe that the ADSA scheme tends to alternate roughly equally between re-allocation and addition of new inputs. To save computational overhead, we consider the simplified *Deterministic Design w/Stepwise Allocation* (DDSA) scheme that deterministically alternates between re-allocation and adding inputs, making $k_n = k_0 + \lceil (n - k_0)/2 \rceil$ also deterministic. Observe that DDSA no longer needs to evaluate the expensive $\mathcal{I}_{ADSA}^{(n)-all}$ and $\mathcal{I}_{ADSA}^{(n)-new}$. A different shortcut is *Fixed Design Stepwise Allocation* (FDSA) which avoids exploration altogether and keeps $k_n = K$ constant by starting immediately with a large initial design $|\mathcal{A}_0| = K$. FDSA thus always uses re-allocation, aiming to grow the number of replicates for inputs in the neighborhood of the contour. We find that the performance of FDSA is quite sensitive to the choice of the initial inputs, and k_0 needs to increase exponentially with dimension d.

5.3.1 Allocation Rule for GP

Proof: [Proof of Proposition 5.3.1] Because the unique inputs are unchanged during the allocation step, comparing $\mathbf{C}^{(n+1)} = \mathbf{K}(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{K}_*(\boldsymbol{\Sigma}^{(n+1)})^{-1}\mathbf{K}_*^T$ to $\mathbf{C}^{(n)} = \mathbf{K}(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{K}_*(\boldsymbol{\Sigma}^{(n)})^{-1}\mathbf{K}_*^T$, the only term that changes is $\boldsymbol{\Sigma}^{(n+1)}$. Minimizing Eq. (5.8) therefore reduces to maximizing

$$(\boldsymbol{\omega}^{(n)})^T\mathbf{K}_*(\mathbf{K} + \tau^2\mathbf{R}^{(n+1)})^{-1}\mathbf{K}_*^T\boldsymbol{\omega}^{(n)} \mapsto \max!. \tag{5.14}$$

Decompose $\Delta\mathbf{R}^{(n)} =: \mathbf{B}^{(n)}\mathbf{B}^{(n)}$. Using the Woodbury Identity,

$$(\boldsymbol{\Sigma}^{(n+1)})^{-1} = (\mathbf{K} + \tau^2(\mathbf{R}^{(n)} - \Delta\mathbf{R}^{(n)}))^{-1} \approx (\boldsymbol{\Sigma}^{(n)})^{-1} + (\boldsymbol{\Sigma}^{(n)})^{-1}\Delta\mathbf{R}^{(n)}(\boldsymbol{\Sigma}^{(n)})^{-1},$$

where the last expression is obtained by dropping the term $\mathbf{B}^{(n)}[\mathbf{K} + \tau^2\mathbf{R}^{(n)}]^{-1}\mathbf{B}^{(n)} \approx \mathbf{0}$ due to $\max_i \Delta\mathbf{R}_{ii}^{(n)} \ll 1$. Therefore, maximizing (5.14) subject to $\sum_{i=1}^{k_n} \Delta r_i^{(n)} = \Delta r^{(n)}$ is equivalent to maximizing

$$\tilde{\mathcal{L}}^{(n+1)}(\Delta\mathbf{R}) = (\boldsymbol{\omega}^{(n)})^T\mathbf{K}_*(\boldsymbol{\Sigma}^{(n)})^{-1}\Delta\mathbf{R}^{(n)}(\boldsymbol{\Sigma}^{(n)})^{-1}\mathbf{K}_*^T\boldsymbol{\omega}^{(n)} + \lambda\left(\Delta r^{(n)} - \sum_{i=1}^{k_n}\Delta r_i^{(n)}\right), \tag{5.15}$$

where λ is a Lagrange multiplier. The first-order optimality conditions are

$$\frac{\partial\tilde{\mathcal{L}}^{(n+1)}}{\partial\Delta r_i^{(n)}} = -\frac{(\boldsymbol{\omega}^{(n)})^T\mathbf{K}_*(\boldsymbol{\Sigma}^{(n)})^{-1}(\boldsymbol{\Sigma}^{(n)})^{-1}\mathbf{K}_*^T\boldsymbol{\omega}^{(n)}}{(r_i^{(n)} + \Delta r_i^{(n)})^2} - \lambda = 0$$

which lead to $r_i^{(n)} + \Delta r_i^{(n)} \propto [(\boldsymbol{\Sigma}^{(n)})^{-1}\mathbf{K}_*^T\boldsymbol{\omega}^{(n)}]_i$, $1 \le i \le k_n$ as in (5.10). ■

Following Liu & Staum [61], we use a pegging procedure [14] to obtain integer-valued $\Delta r_i^{(n)}$. Details are described in Algorithm 5 in the Remark 6. Note that due to the rounding, the added number of replicates $\sum_{i=1}^{k_n}\Delta r_i^{(n)}$ is not exactly $\Delta r^{(n)}$. Moreover, there are several approximations in Proposition 5.3.1 that render $\Delta r_i^{(n)}$ and (5.9) suboptimal: (1) we assume that $\max_{i=1,\dots,k_n}\Delta\mathbf{R}_{ii}^{(n)} \ll 1$; (2) we freeze the weights in (5.8) rather than using $\boldsymbol{\omega}^{(n+1)}$; (3) we round off to integers.

Remark 5 *Similar results about minimizing the look-ahead GP variance of a linear combination* $\boldsymbol{\omega}^T\mathbf{f}$ *appear in [1, 22, 61, 66]. Relative to Ankenman et al. [1] and Chen & Zhou [22], we get rid of all integrals, making* (5.9) *computationally efficient. Liu & Staum [61] use a 3-round allocation which in our experience is not sufficient to fully explore the space; our multi-round approach performs better. The SV-GP algorithm proposed by Ludkovski & Risk [66] relied on in-sample test set* $\mathbf{x}_* = \bar{\mathbf{x}}_{1:k_n}$ *while in our case the test set is different from the existing inputs. Finally, the OCBA algorithm proposed by Chen et al. [20] is designed for optimization, where the allocation rule is derived by maximizing the probability of correctly selecting the optima.*

Also, both SV-GP and OCBA sequentially increase the number of replicates at the pre-fixed inputs, while in our method, we either add new inputs into the current dataset or increase the number of replicates at existing inputs. Thus our algorithm sequentially increases both the numbers of inputs and the simulation calls, while the other two only increase the number of simulation calls.

Last but not least we note that Proposition 5.3.1 can be extended to the heteroscedastic setting by multiplying $\mathbf{U}_i^{(n)}$ *by* $\boldsymbol{\tau}^2(\bar{\mathbf{x}}_{1:k_n})$ *coordinate-wise.*

Remark 6 *Pegging Algorithm 5 [14] is employed to obtain the allocation* $\Delta \boldsymbol{r}^{(n)}_{1,\ldots,k_n}$ *within ADSA as integers.*

Algorithm 5 Pegging Algorithm

Input: $I_0 = \{1, \ldots, k_n\}$, $r = \sum_{i=1}^{k_n} r_i^{(n)}$, $\mathbf{U}^{(n)}$ from Eq. (5.10)
$j \leftarrow 0$.
for all $i \in I_j$ **do**
  $\Delta r_i^{(n)} \leftarrow \frac{\mathbf{U}_i^{(n)}}{\sum_{j=1}^{k_n} \mathbf{U}_j^{(n)}} \times r - r_i^{(n)}$
  if $\Delta r_i^{(n)} \geq 0$ for all $i \in I_j$ **then**
    break
  else
    $I_{j+1} \leftarrow \{i \in I_j : \Delta r_i^{(n)} > 0\}$
    $\Delta r_i^{(n)} = 0$ for $i \notin I_{j+1}$
    $r \leftarrow r - \sum_{i \in I_j, i \notin I_{j+1}} r_i^{(n)}$
    $j \leftarrow j + 1$
  end if
end for
Round all $\Delta r_i^{(n)}, i = 1, .., k_n$ to the nearest integer.
(If $\sum_{i=1}^{k_n} \Delta r_i^{(n)} = 0$, round $\max_{i=1}^{k_n} \Delta r_i^{(n)}$ up to the next integer)

5.3.2 Allocation Rule for t-GP

Our adaptive batching strategies are not limited to the vanilla GP setup. Other metamodels can be straightforwardly substituted as long as they allow to efficiently evaluate the $\mathcal{I}_n$ criteria

and the batch look-ahead variance $s^{(n+1)}(x, r)$. As one instructive example we consider t-GP, which is shown as a good choice in the face of noise misspecification as commonly happens for practical stochastic simulators, cf. Chapter 4. Similar to GP with normally distributed noise, replacing variance of noise ϵ_i in (2.1) with variance $\frac{\tau^2}{r_i^{(n)}}$ leading to the marginal likelihood of $\bar{\mathbf{y}}_{1:k_n}$ in (3.10) as

$$p_{t\mathrm{GP}}(\bar{\mathbf{y}}_{1:k_n}|\bar{\mathbf{x}}_{1:k_n}, \mathbf{r}_{1:k_n}^{(n)}, \mathbf{f}) = \prod_{i=1}^{k_n} \frac{\Gamma((\nu+1)/2)\sqrt{r_i^{(n)}}}{\Gamma(\nu/2)\sqrt{\nu\pi}\tau} \left(1 + \frac{r_i^{(n)}(y_i - f_i)^2}{\nu\tau^2}\right)^{-(\nu+1)/2}. \quad (5.16)$$

We then obtain a Gaussian approximation to the posterior $f(x_*)|\mathcal{A}_n \sim \mathcal{N}(\hat{f}_{t\mathrm{GP}}^{(n)}(x_*), s_{t\mathrm{GP}}^{(n)}(x_*)^2)$ defined in (3.13) and (3.14), where $\mathbf{W}_{t\mathrm{GP}}^{(n)}$ is diagonal with

$$W_{t\mathrm{GP},ii}^{(n)} = -\nabla^2 \log p_{t\mathrm{GP}}(\bar{y}_i|\tilde{f}_i^{(n)}, \bar{x}_i) = (\nu+1)\frac{\nu\frac{\tau^2}{r_i^{(n)}} - (\bar{y}_i - \tilde{f}_i^{(n)})^2}{(\nu\frac{\tau^2}{r_i^{(n)}} + (\bar{y}_i - \tilde{f}_i^{(n)})^2)^2}. \quad (5.17)$$

The approximate step-ahead variance of t-GP defined in Eq. (4.38) is then changed into:

$$s_{t\mathrm{GP}}^{(n+1)}(x_{k_n+1}, r_{k_n+1}^{(n)})^2 \simeq s_{t\mathrm{GP}}^{(n)}(x_{k_n+1})^2 \cdot \frac{\frac{\tau^2}{r_{k_n+1}^{(n)}}\frac{\nu+1}{\nu-1}}{\frac{\tau^2}{r_{k_n+1}^{(n)}}\frac{\nu+1}{\nu-1} + s_{t\mathrm{GP}}^{(n)}(x_{k_n+1})^2}. \quad (5.18)$$

We replace Eq. (4.27) with (5.18) to obtain $\mathcal{I}$ acquisition functions for t-GP. To calculate allocation rule for t-GP with ADSA and DDSA, we assume $(\bar{y}_i - \tilde{f}_{t\mathrm{GP}}^{(n)}(\bar{x}_i))^2 = \frac{\tau^2}{r_i^{(n)}}$ and $\tilde{f}_{t\mathrm{GP}}^{(n)}(\bar{x}_i) = \tilde{f}_{t\mathrm{GP}}^{(n+1)}(\bar{x}_i)$. Similar results to Proposition 5.3.1 are obtained.

To implement ADSA and DDSA for t-GP we need (i) the analogue of Proposition 5.3.1 for the allocation rule $\Delta\mathbf{r}_{1:k_n}^{(n)}$ over the existing inputs $\bar{\mathbf{x}}_{1:k_n}$; (ii) the look-ahead variance $s^{(n+1),new}(x_*)$ conditional on adding a new input; (iii) look-ahead variance $s^{(n+1),all}(x_*)$ conditional on allocating $\Delta\mathbf{r}_{1:k_n}^{(n)}$. For all these tasks, the non-Gaussian likelihood (3.10) underlying t-GP calls for further approximations provided in the following three Lemmas.

Lemma 5.3.2 *The allocation rule is like in Proposition 5.3.1 but relies on*

$$\tilde{\mathbf{U}}_{t\mathrm{GP}}^{(n)} = (\tilde{\boldsymbol{\Sigma}}_{t\mathrm{GP}}^{(n)})^{-1}\mathbf{K}_*^T\boldsymbol{\omega}^{(n)}, \qquad \textit{with} \quad \tilde{\boldsymbol{\Sigma}}_{t\mathrm{GP}}^{(n)} := \left(\mathbf{K} + \frac{\nu+1}{\nu-1}\tau^2\mathbf{R}^{(n)}\right). \tag{5.19}$$

Proof: [Proof of Lemma 5.3.2] For t-GP, the noise matrix $\tau^2\mathbf{R}^{(n)}$ in Eq. (5.2) is replaced with $(\boldsymbol{W}_{t\mathrm{GP}}^{(n)})^{-1}$. By definition, $\boldsymbol{W}_{t\mathrm{GP}}^{(n)}$ is a diagonal matrix with entries

$$\begin{aligned} W_{ii}^{(n)} &= (\nu+1)\frac{\nu\frac{\tau^2}{r_i^{(n)}} - (\bar{y}_i - \tilde{f}_i^{(n)})^2}{(\nu\frac{\tau^2}{r_i^{(n)}} + (\bar{y}_i - \tilde{f}_i^{(n)})^2)^2} \\ &\approxeq (\nu+1)\frac{\nu\frac{\tau^2}{r_i^{(n)}} - \frac{\tau^2}{r_i^{(n)}}}{\left(\frac{\tau^2}{r_i^{(n)}} + \nu\frac{\tau^2}{r_i^{(n)}}\right)^2} = \frac{(\nu-1)r_i^{(n)}}{(\nu+1)\tau^2} := \tilde{W}_{ii}^{(n)}, \end{aligned}$$

where we use the approximation $(y(x_i) - \tilde{f}(x_i))^2 \simeq \tau^2$ or $(\bar{y}_i - \tilde{f}(\bar{x}_i))^2 \simeq \tau^2/r_i^{(n)}$. Hence, $(\boldsymbol{W}_{t\mathrm{GP}}^{(n)})^{-1} \simeq (\tilde{\boldsymbol{W}}_{t\mathrm{GP}}^{(n)})^{-1} = \frac{\nu+1}{\nu-1}\tau^2\mathbf{R}^{(n)}$ and hence the covariance matrix $\mathbf{C}_{t\mathrm{GP}}^{(n)}$ of $f(\mathbf{x}_*)$ is approximated as

$$\begin{aligned} \mathbf{C}_{t\mathrm{GP}}^{(n)} &= \mathbf{K}(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{K}_*\left(\mathbf{K} + (\mathbf{W}_{t\mathrm{GP}}^{(n)})^{-1}\right)^{-1}\mathbf{K}_*^T \\ &\simeq \mathbf{K}(\bar{\mathbf{x}}_*, \bar{\mathbf{x}}_*) - \mathbf{K}_*(\tilde{\boldsymbol{\Sigma}}_{t\mathrm{GP}}^{(n)})^{-1}\mathbf{K}_*^T, \end{aligned} \tag{5.20}$$

where $\tilde{\boldsymbol{\Sigma}}_{t\mathrm{GP}}^{(n)})$ is from (5.19). The rest of the proof then matches Proposition 5.3.1 to obtain $\tilde{\boldsymbol{U}}_{t\mathrm{GP}}^{(n)}$. ■

Lemma 5.3.2 implies that we may proceed exactly like for regular GP model, after boosting up τ^2 to $(\nu+1)/(\nu-1)\tau^2$. Combined with Algorithm 5 this yields $\Delta\boldsymbol{r}_{1:k_n}^{(n)}$.

Unlike in the Gaussian case where $\boldsymbol{\Sigma}^{(n+1)}$ depends only on $\mathbf{R}^{(n+1)}$, for t-GP the future $\boldsymbol{W}_{t\mathrm{GP}}^{(n+1)}$ depends on $\bar{\mathbf{y}}_{1:k_n}$ (because it depends on $\tilde{\mathbf{f}}_{t\mathrm{GP}}$). We therefore need an approximation $\hat{\boldsymbol{W}}_{t\mathrm{GP}}^{(n+1)}$.

Lemma 5.3.3 *The look-ahead variance at x_* conditional on allocating $\Delta r^{(n)}$ simulations to a new input $\bar{x}_{k_n+1}$ is approximately given by*

$$\tilde{s}^{(n+1),new}_{t\mathrm{GP}}(x_*)^2 \approxeq s^{(n)}_{t\mathrm{GP}}(x_*)^2 - \frac{v^{(n)}_{t\mathrm{GP}}(x_*, \bar{x}_{k_n+1})^2}{\frac{(\nu+1)\tau^2}{(\nu-1)\Delta r^{(n)}} + s^{(n)}_{t\mathrm{GP}}(\bar{x}_{k_n+1})^2}. \tag{5.21}$$

Lemma 5.3.3 is obtained directly from Eq. (4.27). We use (5.21) to obtain $\mathcal{I}^{(n),new}_{ADSA}$.

Finally, to obtain $\mathcal{I}^{(n),all}_{ADSA}$ we define

$$\tilde{W}^{(n+1),all}_{ii} := (\nu+1)\frac{\nu\frac{\tau^2}{r_i^{(n+1)}} - (\bar{y}_i^{(n)} - \tilde{f}^{(n)}_{t\mathrm{GP}}(\bar{x}_i))^2}{\left((\bar{y}_i^{(n)} - \tilde{f}^{(n)}_{t\mathrm{GP}}(\bar{x}_i))^2 + \nu\frac{\tau^2}{r_i^{(n+1)}}\right)^2}, \tag{5.22}$$

based on the approximation $(\bar{y}_i^{(n+1)} - \tilde{f}^{(n+1)}_{t\mathrm{GP}}(x_i))^2 = (\bar{y}_i^{(n)} - \tilde{f}^{(n)}_{t\mathrm{GP}}(x_i))^2$. This yields

Lemma 5.3.4

$$\tilde{s}^{(n+1),all}_{t\mathrm{GP}}(x_*) \approxeq K(x_*, x_*) - \mathbf{K}_*\left(\mathbf{K} + (\tilde{\mathbf{W}}^{(n+1),all}_{t\mathrm{GP}})^{-1}\right)^{-1}\mathbf{K}_*^T. \tag{5.23}$$

5.4 Synthetic Experiments and Case Study

5.4.1 Synthetic Experiments and Computational Implementation Details

In this section we consider three case studies, employing rescaled *Branin-Hoo* ($d = 2$) and *Hartman* ($d = 6$) functions defined in Table 4.3. The first two case studies are in 2-D and use the Branin-Hoo as the true response with two noise choices: (i) Gaussian $\epsilon \sim \mathcal{N}(0, 1)$; and (ii) heteroscedastic Student-t: $\epsilon \sim t_{6-4x^1}(0, (0.4(4x^1 + 1))^2)$. The second case is to test the influence of noise misspecification. The third case study is in 6-D using the Hartman function and noise $\epsilon \sim \mathcal{N}(0, 1)$. The noise variance is taken to be known (i.e. $\tau = 1$) in the first and

Table 5.1: Parameter set-ups for 2-D *Branin-Hoo* experiments and 6-D *Hartman* experiments.

PARAMETER	*Branin-Hoo*	*Hartman*
N_T	2000	6000
k_0	20	60
r_0	10	10
M	500	1000
$\mathbf{r}_L$	$[5, 10, 15, 20, 30, 40, 50, 60, 80, 100, 140, 180, 240, 300]$	
$\mathcal{R}$	$[5, 200]$	$[5, 300]$
T_{sim}	0.01	0.05
c_{batch}	5	1.67
$c_{over}(n)$	$\boldsymbol{\theta} = [0.137, 8.15 \times 10^{-4}, 1.99 \times 10^{-6}]$	

third case studies. It is fitted (as an unknown constant) along with $\boldsymbol{\vartheta}$ for the experiments with Student-t simulation noise.

We use FB with $r \equiv 10$ as a baseline, and compare the performance of MLB, RB, ABSUR, DDSA and ADSA. Performance is based on the error rate $\mathcal{ER}$ in (3.34), i.e. evaluating (numerically, using a testing set of size M) the symmetric difference between the true and estimated level set. This is done at a fixed simulation budget N_T, i.e. each scheme is run until N_{k_n} reaches that budget. Note that the resulting number of DoE rounds will vary scheme-by-scheme and is denoted as k_T. Recall that N_n, k_n are indexed by the DoE sequential iterations, while N_T, k_T are indexed by total budget consumed. Table 5.1 provides further details about the parameters specific to each scheme.

Whenever we use MCU we follow the recipe in Chapter 4 and set $\gamma^{(n)} = IQR(\hat{f}^{(n)})/3Ave(s^{(n)})$ which keeps both terms in (4.14) approximately comparable as n changes. For MLB, we initialize γ as the average standard deviation $Ave(s^{(k_0)}(\bar{x}_{1:k_0}))$ and take $\eta = 0.6$. For RB we use the same initial γ but with $\eta = 0.95$. For ABSUR, we recommend $\underline{r}$ of 5 or 10, and $\bar{r} = 0.05N_T$, i.e. 5% of the total budget N_T. The coefficients $\boldsymbol{\theta}$ in the quadratic overhead function $c_{over}(n)$ in (5.5) are pre-tuned via a linear least squares regression with the given simulator and hardware setup.

5.4.2 Algorithm Performance

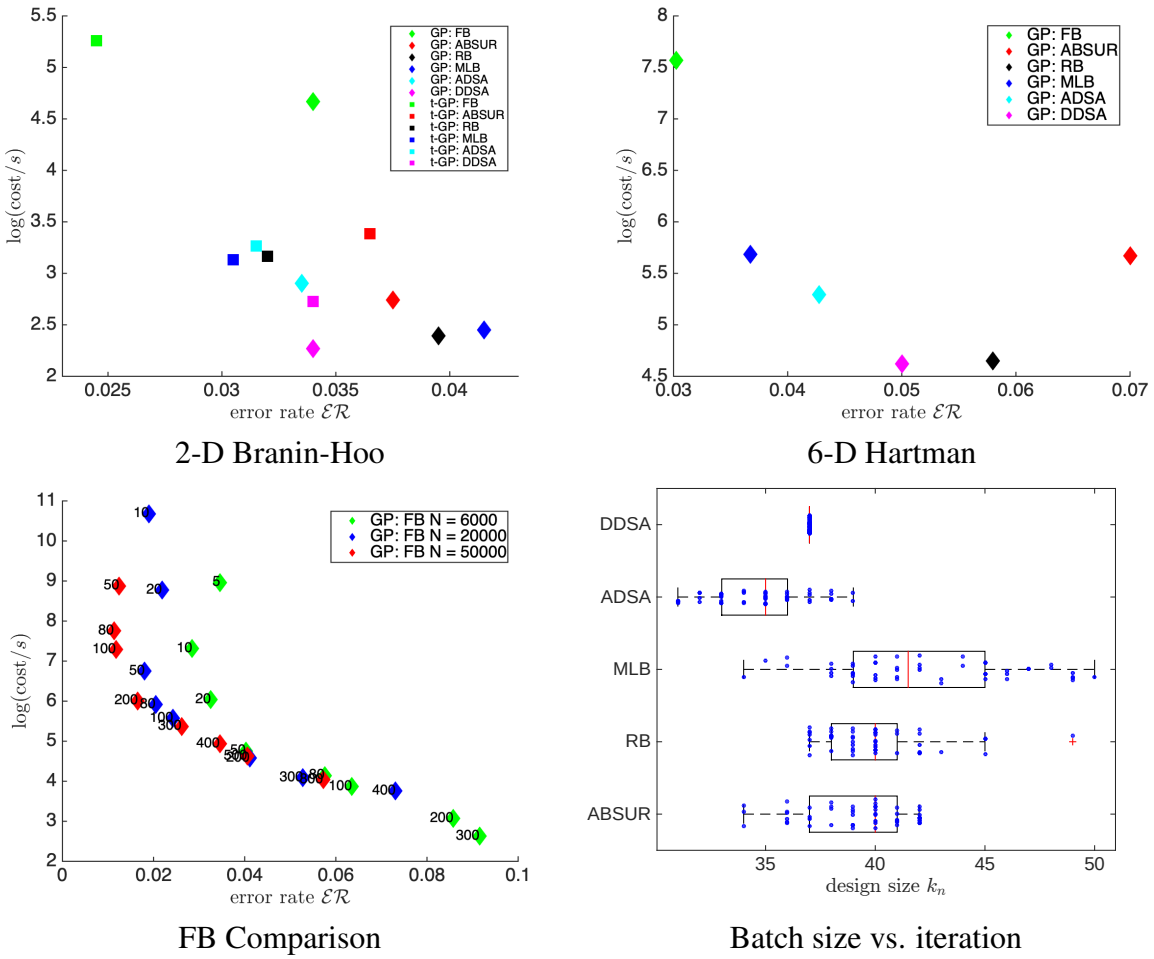


Figure 5.1: Running time and ultimate error rate $\mathcal{ER}_T$ across different schemes. *Upper Left:* 2-D Branin-Hoo with Student-t noise and budget $N_T = 2000$. *Upper Right:* 6-D Hartman function with Gaussian noise and $N_T = 6000$. *Lower Left:* 6-D Hartman function with Gaussian noise for FB with different values of r. We show median results from 50 runs of each method, with color identifying the scheme and diamonds/squares identifying the GP/t-GP metamodels. *Lower Right:* distribution of design size k_T corresponding to $N_T = 2000$ across 50 algorithm runs.

Figure 5.1 and Table 5.2 show the trade-off between the error rate $\mathcal{ER}$ from (3.34) and the running time. Since we desire fast and accurate schemes, there is a Pareto frontier going from top-left to bottom-right. In the 2-D case study (shown in the upper left panel in Figure 5.1), we see that the most accurate scheme is t-GP with FB, while the fastest is GP with DDSA.

Other efficient schemes are t-GP with MLB which is arguably the best (the second fastest among t-GPs, and the second most accurate). In 6-D ABSUR works poorly, probably due to under-performance of the cSUR criterion; In Section 4.3 we showed that MCU appears to be empirically better for this 6-D Hartman function. Another reason is that cSUR converges in a slower rate, see plot (b) in Figure 5.2: cSUR takes $N_T \approx 30000$ simulations to achieve a comparably small error rate $\mathcal{ER}$. However, in Figure 5.1, $N_T = 6000$ for 6-D experiments. The best choice are MLB and ADSA, as DDSA and RB gain some speed but only at significant increase in $\mathcal{ER}$.

Looking at the running times, we see that there are major gains from adaptive batching; the baseline FB scheme takes almost 10 times longer to run than adaptive batching designs. Fixed batching generally performs well in terms of $\mathcal{ER}$ (as it ends up being more exploratory) but practically those gains are crowded out by the huge gains in computational efficiency. Among the five proposed schemes, MLB and ADSA tend to produce lower $\mathcal{ER}$ with a significant reduction in computational time, especially in 6-D experiments.

In the 6-D case study, performance of FB scheme with different replicates r_n for $N_T = 6000$, $N_T = 20000$ and $N_T = 50000$ is compared (shown in the lower left panel in Figure 5.1). For all budgets N_T, the running time is higher and the error rate $\mathcal{ER}$ is approximately lower for lower r_n, and vice versa. However, we observe that when the replicate is extremely low, for example, r_n is smaller than 20 for $N_T = 6000$, the error rate $\mathcal{ER}$ is not increasing with replicate r_n. Basically as r_n is small enough, k_n is large enough and the space is fully explored. At this time, we do not gain a lot of accuracy by adding more new inputs. For each N_T, there is an "optimal" value of r_n for FB such that the error rate $\mathcal{ER}$ is lowest: for $N_T = 6000$, the optimal r_n is 10; for $N_T = 20000$, the optimal r_n is 50, and for $N_T = 50000$, the optimal r_n is 100.

The plot (a) of Figure 5.2 shows the $\log$ error rate $\mathcal{ER}$ as a function of running time for FB, ABSUR, RB, MLB, ADSA and DDSA for 2-D *Branin-Hoo* experiments with Student-t noise and 6-D *Hartman* experiments. For FB, we stopped at $N_T = 6000$ due to prohibitive running

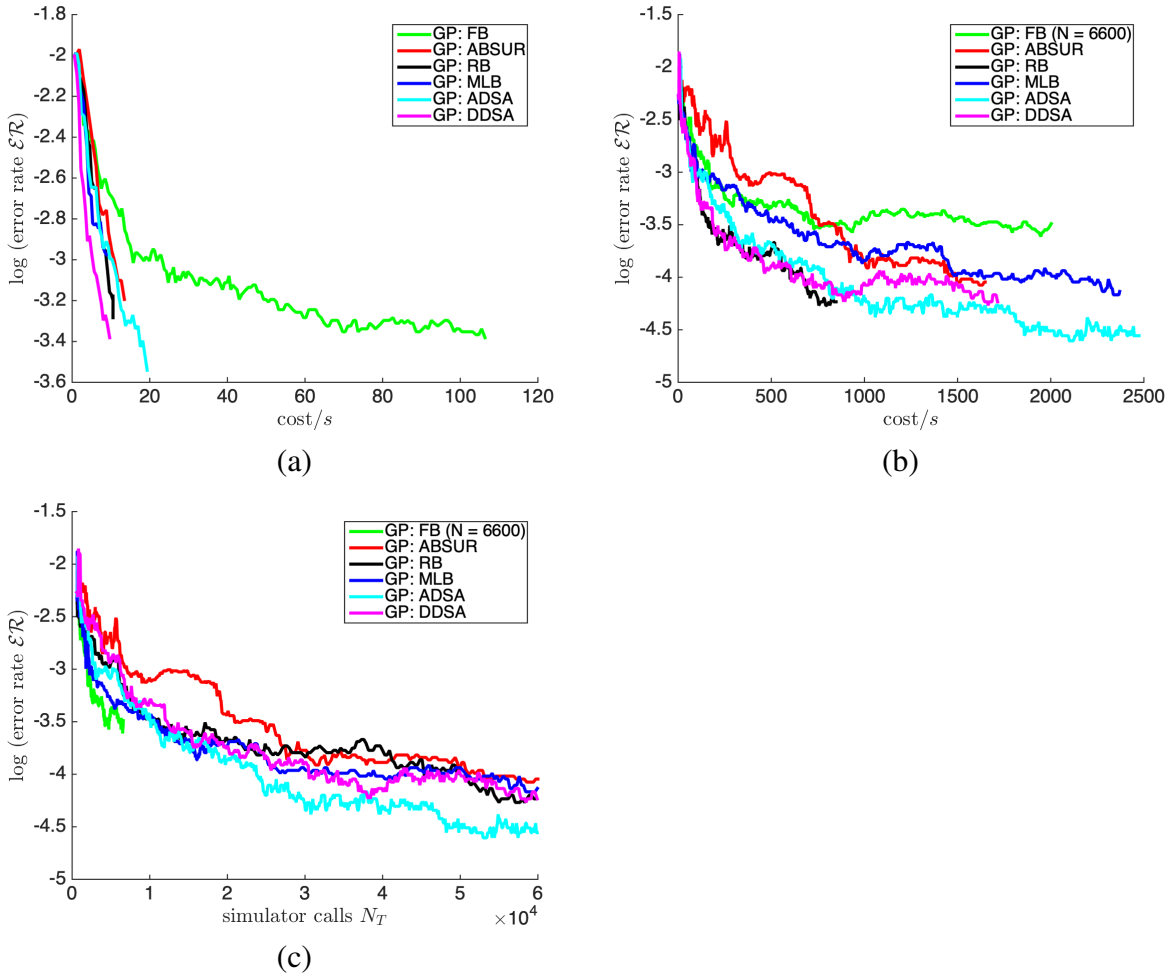


Figure 5.2: Log error rate $\log \mathcal{ER}_t$ as a function of running time t for FB (green), ABSUR (red), RB (black), MLB (blue), ADSA (cyan) and DDSA (magenta) for 2-D *Branin-Hoo* experiments with Student-t noise (plot (a)) with $N_T = 2000$ and 6-D *Hartman* with Gaussian noise (plot (b)) with $N_T = 60000$. Log Error rate $\log \mathcal{ER}_t$ as a function of simulator calls N_t for 6-D *Hartman* experiments (plot (c)). The FB algorithm is stopped at $N_t = 6000$ since computation is too slow.

times for designs. We observe that all the schemes reduce the error rate $\mathcal{ER}$ at a faster rate than FB. In the early stage, RB and DDSA are the fastest, and ABSUR is the slowest. However, as time proceeds, ADSA keeps reducing the error rate $\mathcal{ER}$ and converges to a smaller $\mathcal{ER}$ than other algorithms with a fixed total simulator calls N_T. ADSA usually takes slightly longer time to end up all the simulations. In plot (c), we plot the log error rate $\mathcal{ER}$ as a function of simulator calls N_n for the 6-D experiments. We observe that besides FB, MLB reduces the error rate $\mathcal{ER}$

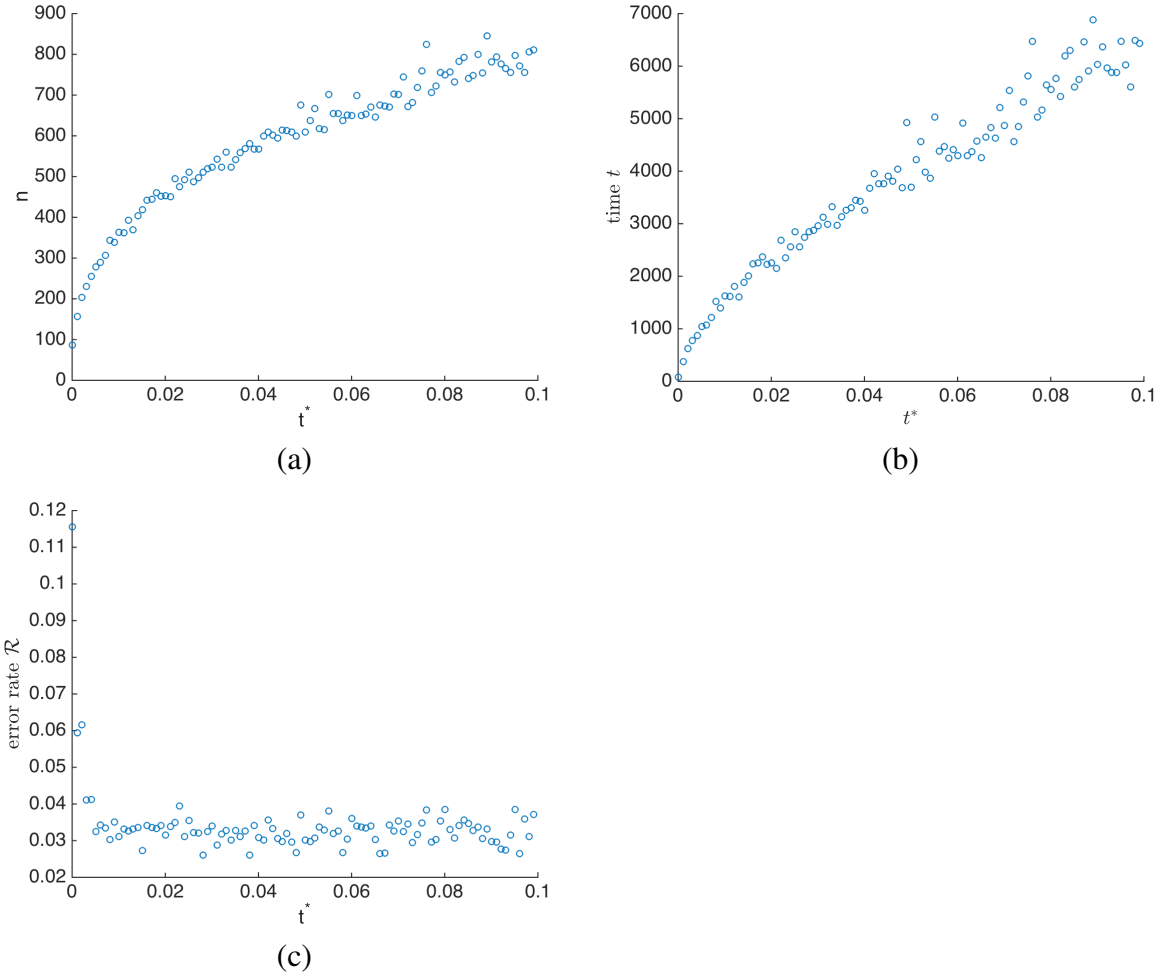
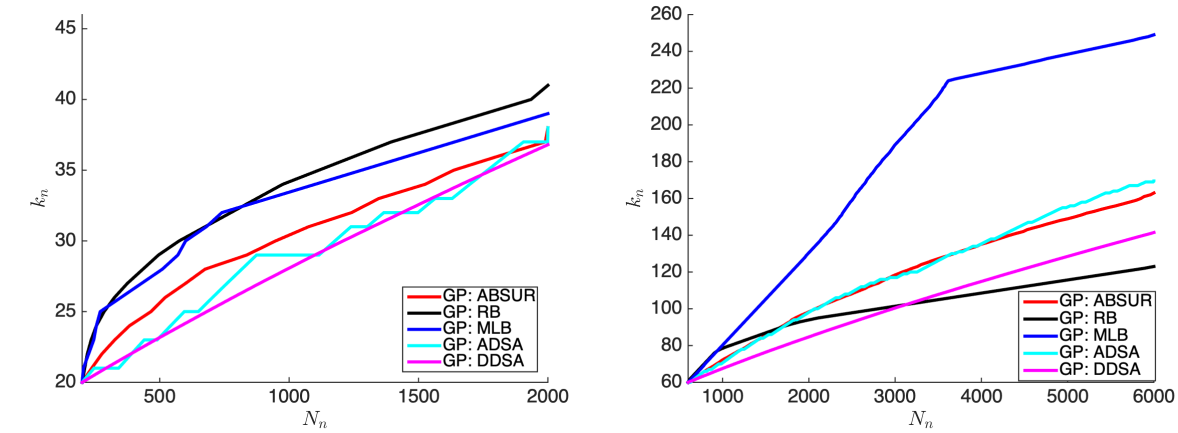


Figure 5.3: Design size k_T, timing cost t and error rate $\mathcal{ER}$ as a function of t^* for ABSUR in 6-D *Hartman* with Gaussian noise.

at the fastest rate when $N_n < 600$, and otherwise, ADSA is the fastest. ADSA shines in the later stage of sequential development of DoE, since it needs enough "candidate inputs" to calculate the allocation rule. In conclusion, ADSA is the most accurate algorithm given a large enough cost t or simulator calls N_T, and MLB is the most accurate algorithm when N_T is small. Results are consistent with those observed in Figure 5.1.

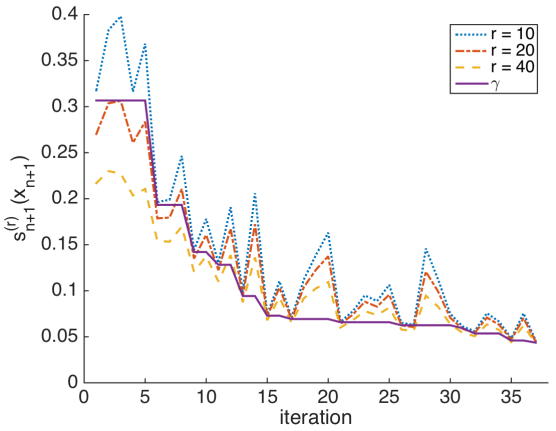
We also investigate the effect of t^* at performance of ABSUR in plot (a)-(c) in Figure 5.3. Basically the design size k_T and the running time t will increase with t^* as shown in plot (a) and (b). Intuitively, larger t^* represents more expensive simulation, and thus ABSUR picks a smaller r at each step to increase the information gain per unit of running time defined in (5.4). In plot (c), we observe that the error rate $\mathcal{ER}$ explodes when the value of t^* is too small (smaller than 0.01), and keeps stable for other values of t^*. Therefore, the performance of ABSUR is not sensative to value of t^* as long as the space is well explored. We set $t^* = 0.01$ for 2-D *Branin-Hoo* and $t^* = 0.05$ for 6-D *Hartman*. The actual value of t^* depends on the simulation setups and the machine configurations.

Recall that GP complexity is $\mathcal{O}(k_n^3)$, so that the design size $k_n = |\mathcal{A}_n|$ is the primary driver of computational efficiency. In the baseline FB scheme, $r^{(n)} \equiv r$ is constant so that $k_n = N_n/r$ grows linearly in simulator budget N_n. A key aim of adaptive batching is to achieve *sub-linear* growth of k_n i.e. $k_n/N_n \to 0$ as n grows so that $r^{(n)}$ keeps getting larger as we develop the DoE. Upper panel of Figure 5.4 plots k_n as a function of N_n for 2-D and 6-D experiments. As desired, we observe a generally concave shape, which is approximately of square-root shape. The stair-case shape of k_n for ADSA is due to the adaptive re-allocation of new simulations which allow to increase N_n without changing k_n at some steps. We note that RB and ADSA achieve the most concave shape and hence would be the fastest for very large N_n.

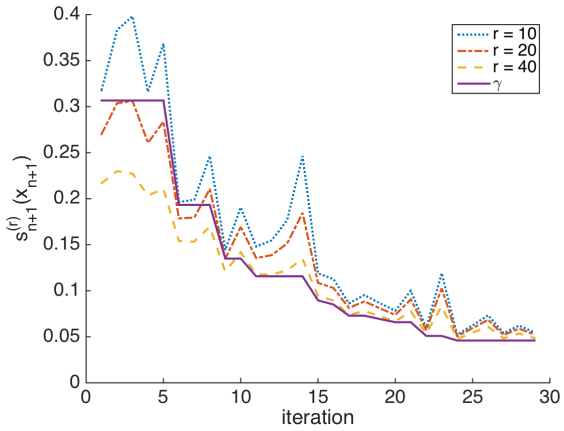


2-D Branin-Hoo with Student-t noise

6-D Hartman with Gaussian noise



MLB



RB

Figure 5.4: *Upper panel:* The design size k_n as a function of simulator calls N_n. *Lower panel:* Comparison of the standard deviation $s^{(n+1)}(\bar{x}_{n+1}, r)$ and γ as a function of iteration n for 2-D *Branin-Hoo*.

5.4.2.1 Comparing Designs

To drill down into the designs obtained from different approaches, Figure 5.1 and Figure 5.5 visualize the adaptively batched designs produced for the 2-D Branin-Hoo experiment with Student-t noise. The lower right panel in Figure 5.1 displays the resulting design size k_T with simulation budget of $N_T = 2000$. Recall that besides FB and DDSA, design sizes of all other schemes vary across algorithm runs (i.e. k_T depends on the particular realizations $y_{1:N_T}$), so that k_T is a random variable; in the plot we visualize its boxplot across 50 runs of each scheme.

The smallest designs are obtained from ADSA (31-39 unique inputs). DDSA produces exactly $k_T = 37$ unique inputs. Recall that DDSA alternates between adding a new site and re-allocating to existing sites, while ADSA does the same adaptively; in this case we find that slightly more than half the time re-allocation is preferred. The design size k_n for ABSUR is slightly larger at 34-42. The value of k_T for RB varies from 37 to 45, while for MLB has the greatest number of unique inputs, ranging from 34 to 50. Given $N_T = 2000$ the above implies that the schemes average about $Ave(r^{(n)})$ =40-60 replicates per site.

Lower panel of Figure 5.4 visualizes how MLB and RB select value of r by comparing the one-step ahead standard deviation $s^{(n+1)}(\bar{x}_{n+1}, r)$ and the threshold γ. To simplify the plot, the levels for batch size r is set to be $\mathbf{r}_L = \{10, 20, 40\}$. For MLB, at each step, we choose the largest r such that the standard deviation $s^{(n+1)}(\bar{x}_{n+1}, r)$ is above the threshold γ. We observe that the value of batch size r converges to the highest level $r = 40$ in approximately $n = 13$ iterations, and flips back to $r = 10$ or $r = 20$ afterwards at several steps (for example, $n = 17$ and $n = 21$). RB chooses between keeping the current level or moving forward to the next level of $\mathbf{r}_L$. If the one-step ahead standard deviation of both the current level and the next level is below the threshold γ, then we decrease the value of γ (see, for example, in step 6). Compared with MLB, RB takes less iterations to converge to the highest level $r = 40$, with approximately $n = 10$ iterations, and keeps at $r = 40$ afterwards. Therefore, RB ends up with a smaller design size k_T compared with MLB.

The left panel of Figure 5.5 shows the replication level $r^{(n)}$ as a function of design size k_n for a typical run of schemes from Section 5.2.5, illustrating how replication is increased sequentially. Methods that raise $r^{(n)}$ faster end up with smaller design size k_T. ABSUR increases $r^{(n)}$ the fastest, with MLB having a similar pattern. With RB $r^{(n)}$ grows slower, implying that RB builds designs with more unique inputs.

The right panel of Figure 5.5 visualizes the replication of a representative ADSA run which has the option to add new inputs or re-allocate to existing ones. We show the sequential growth

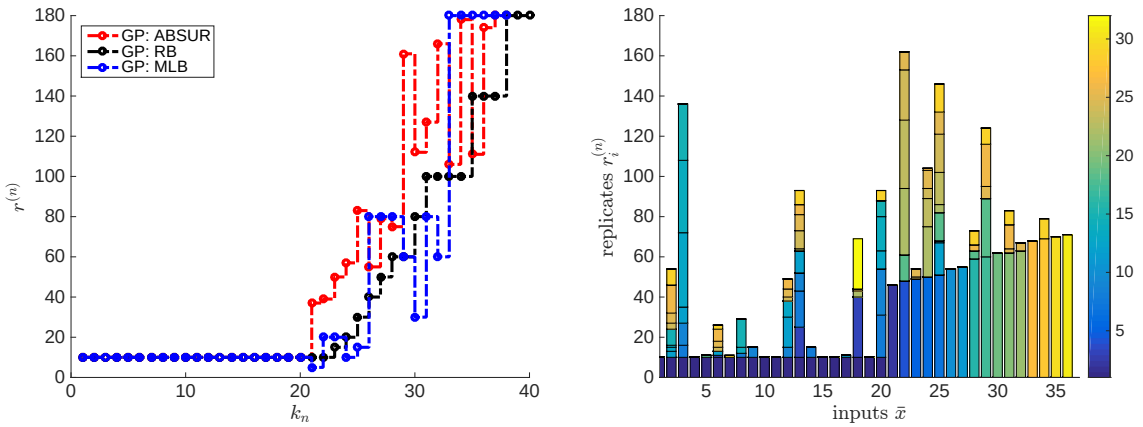


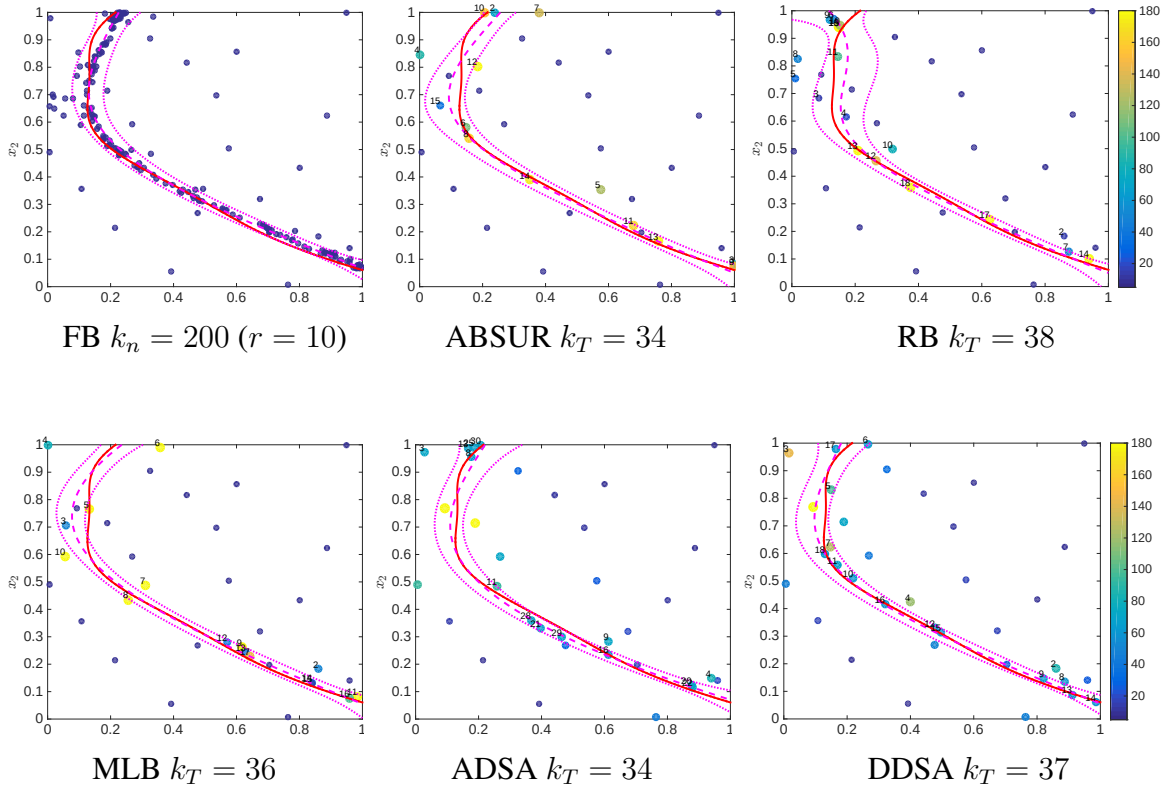
Figure 5.5: Visualizing adaptive batching for 2-D *Branin-Hoo* experiments with Student-t distributed noise. *Left:* number of replicates $r^{(n)}$ as a function of design size k_n for the schemes of Section 5.2. *Right:* evolution of $r_i^{(n)}$ for ADSA designs $\bar{\mathbf{x}}_{1:k_n}$. The total $r_i^{(N)}$ is decomposed into $\Delta r_i^{(n)}$ for $n = 1, \ldots, k_T$ with each Δr color-coded by round n.

of $r_i^{(n)}$ through a stack histogram: the x-axis represents the unique inputs $\mathbf{x}_i$ as picked by the algorithm and the vertical stacks represent $\Delta r_i^{(n)}$, color-coded by the round n when they were added. We observe that only 10 out of the $n_0 = 20$ original inputs are revisited, and generally about half of the inputs are used in more than one round. At the same time, some inputs, such as $\bar{x}_{13}, \bar{x}_{20}, \bar{x}_{25}$ are visited in numerous rounds.

Figure 5.6 shows the estimated zero-contour $\partial\hat{S}$ with its 95% posterior credible band $\partial S^{(\pm 0.95)} = \{x : \hat{f}^{(n)}(x) \pm 1.96 s^{(n)}(x) = 0\}$ at $N_T = 2000$ in the 2-D test case with Gaussian noise. As expected, all schemes start by exploring the input space using a few replicates and then primarily sample in the target region around the level set, with increasing replication. Comparing the first four plots, we find that the ABSUR is more efficient than RB and MLB, concentrating at the zero-contour faster and simultaneously faster ramp-up of $r^{(n)}$. In the plot, this happens already after just half-a-dozen steps. In contrast, RB takes about a dozen steps to explore with correspondingly low $r^{(n)}$'s. Although MLB also ramps up r_n quickly, it then steps back and forth between low and high replication levels, resulting in a slightly larger k_T than ABSUR. ADSA and DDSA perform similarly. One observation is that they select similar inputs

to allocate the extra simulator calls. For example the initial inputs close to the left and bottom edge all get more replicates r_n via reallocation in ADSA and DDSA. Another example is the initial input in the upper left corner of the space gets the most replicates (color yellow) compared with all other inputs for both algorithms. Across the DoE rounds, ADSA chooses to reallocate budget in approximately 54% of them, so that $k_T = 0.54N_T/\Delta r$. Therefore, the value of k_T is approximately the same for ADSA and DDSA. Some of the design differences can be attributed to the different behavior of the underlying heuristics MCU and cSUR. Indeed MCU tends to over-emphasize sampling around the zero-contour, while cSUR is more exploratory and tends to place a few inputs right at the edge of the input domain (upper left corner and lower right corner in the plot with ABSUR). The aggressiveness of MCU generates more accurate estimates $\partial\widehat{S}$ even if the posterior uncertainty is higher (CI band is wider) sometimes.

To conclude, the performance of FB is sensitive to value of replicates r_n. Basically, FB with higher r_n is more efficient. However, the effect of higher r_n on accuracy of the algorithm may change in different directions as the total budget N_T changes. The question is how to pick a value for batch size r_n to strike a balance between the efficiency and accuracy. For different budget N_T, the "optimal" value of r_n varies. Therefore, in online experiments where N_T is not provided initially or there is an infinite budget N_T, it is impossible to pick r_n for FB. Adaptive batching designs stand out perfectly. Instead of picking r_n manually at the start of sequential design, adaptive batching algorithms self-adaptively pick the current "optimal" r_n during sequential design. Among all adaptive batching designs, DDSA and RB are the most efficient algorithms, while ADSA ends up with the most accurate estimate in most cases. For lower dimension experiments or larger N_T, DDSA reaches similar or even better error rate $\mathcal{ER}$ compared with ADSA, while in higher dimension experiments or smaller N_T, results obtained with ADSA are significantly better than DDSA.



FB $k_n = 200$ $(r = 10)$ ABSUR $k_T = 34$ RB $k_T = 38$

MLB $k_T = 36$ ADSA $k_T = 34$ DDSA $k_T = 37$

Figure 5.6: GP fits $f|\mathcal{A}_{k_T}$ and designs for 2-D Branin-Hoo case study with Gaussian noise. The dashed lines are the estimated posterior zero-contours $\hat{f}^{(N)}(x) = 0$ to be compared to the true contour (solid line). The dotted lines are the corresponding 95% credible intervals. The labels indicate the order of the inputs $\bar{x}_i, i = 1, \ldots k_n$ and the respective color/size are proportional to replication level $r^{(n)}$. Design sizes k_T vary across the schemes.

5.4.3 Application to Optimal Stopping

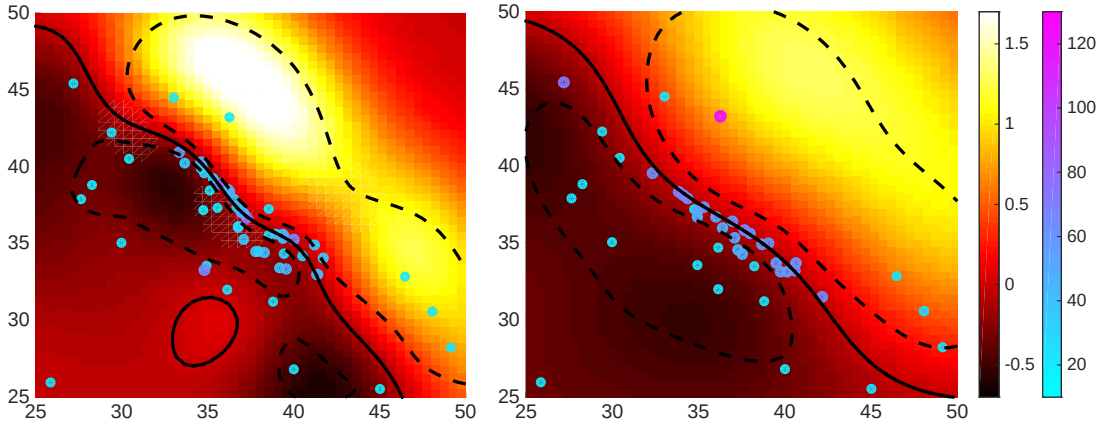
As a fourth and final case study, we consider an application of contour finding for determining the optimal exercise policy of a Bermudan financial derivative [64], with parameters matching the test cases in Section 4.4:

2-D average Put option: $h_{Put}(t, \boldsymbol{x}) = e^{-rt}(K - x^1 - x^2)_+$;

3-D Max-Call option: $h_{Call}(t, \boldsymbol{x}) = e^{-rt}(\max(x^1, x^2, x^3) - K)_+$.

These settings have very low signal-to-noise ratio, and non-Gaussian heteroscedastic noise, so $N_T \gg 10^3$ is imperative. Detailed experimental set-ups are specified in Table 5.3.

Table 5.4 shows the performance of different designs/models. In the 2-D setting the best performing scheme is ABSUR. We obtain savings of 70% in computation time compared to the baseline FB scheme. For the 3-D Max Call, ADSA achieves the highest payoff, and at a fraction ($\sim 1/10$th) of time. RB and MLB lead to basically the same payoff as ADSA, but with a higher computation cost. DDSA leads to slightly lower payoffs and takes approximately the same time compared with ADSA. ABSUR takes half the time of ADSA, leading to a lower payoff. The design size k_T is consistent with results in Figure 5.5: MLB yields the largest k_T, while DDSA and ADSA yield the most compact designs. We also observe that for the 3-D Max Call, the computation cost for ADSA is smaller than DDSA. One interpretation is that large amount of simulations for one new input is much cheaper than simulating and updating the observations for several different existing inputs, since the ADSA ends up with a much larger design size k_T compared with DDSA in this case.



ABSUR $k_T = 60$ ADSA $k_T = 45$

Figure 5.7: GP fits $f^{(k_T)}(t, \cdot)$ and designs $\mathcal{A}$ for 2-D average put option example at $t = 0.6$ and $N_T = 2000$. *Left panel:* ABSUR; *right:* ADSA. The solid lines are the estimated exercise boundary $\hat{f}^{(k_T)}(t, \boldsymbol{z}) = 0$ and the dashed lines are the corresponding 95% credible intervals. The scatter plot is the design $\mathcal{A}_{k_T}$ color-coded by replicate counts $r_i, i = 1, \ldots, k_T$.

Figure 5.7 shows the GP fits $\hat{f}(t, \boldsymbol{x})$ for ABSUR and ADSA for the 2-D Put case study at $t = 0.6$. The desired zero-level contour goes from NW to SE and due to the chosen setting should be symmetric about the $x^1 = x^2$ line. We see that both strategies select inputs around the contour; ABSUR is somewhat more aggressive and yields narrower credible interval for $\partial\hat{S} = \{\hat{f}^{(k_T)} = 0\}$. ABSUR uses more design sites $k_T(ABSUR) = 60 > k_T(ADSA) = 45$ and has a flatter distribution of replication counts. In contrast, ADSA uses up to $r^{(n)} = 120$ replicates.

In both 2-D Average Put and 3-D Max Call examples, all schemes have similar payoff but significantly different computation cost. With batch size $r = 20$ for 2D and $r = 30$ for 3D, FB takes 20 times longer time in the worst compared with adaptive batching design schemes, although its payoff is slightly (but not significantly) higher. Among all adaptive batching designs, ADSA has the best performance, with the highest payoff in most cases and comparably efficient computation. ADSA, RB and MLB achieve similar payoff, while ADSA takes half of the computation time compared with RB and MLB in higher dimensional case. The computation cost of ADSA is approximately 1.5-2 times compared with ABSUR and DDSA (besides the 3D case), while the performance of ADSA is more stable for small N_T or more complicated cases, and the resulting payoff is higher. Also, in higher dimensional cases, ADSA often ends up with larger design size k_T compared with DDSA. Intuitively, this is consistent to the conclusions from Chapter 4 that designs like ICU that favors space-filling work better in higher dimensional cases. ADSA adaptively chooses from exploration and exploitation, while DDSA deterministically alternates between them. Consequently, ADSA adjust to higher dimension with more exploration and achieves better results. When the design size k_T is large and the simulation overhead is not negligible, reallocation might be more expensive than adding new inputs, and ADSA is a better choice over DDSA in terms of both accuracy and efficiency.

Table 5.2: Scheme performance across the three synthetic case studies. Results are means ($\pm$ standard deviations) from 50 runs of each combination of metamodel and batching scheme.

DESIGN	MODEL	ERROR RATE $\mathcal{ER}_T$	TIME/S	AVE k_T
2-D Branin-Hoo WITH $\epsilon \sim \mathcal{N}(0,1)$				
FB	GP	0.019 ± 0.005	119 ± 6	200
ABSUR	GP	0.021 ± 0.007	10 ± 2	35
RB	GP	0.021 ± 0.008	8 ± 1	39
MLB	GP	0.018 ± 0.008	9 ± 2	38
ADSA	GP	0.020 ± 0.008	14 ± 1	34
DDSA	GP	0.022 ± 0.007	8 ± 1	37
2-D Branin-Hoo WITH $\epsilon \sim t_{6-4x^1}(0, (0.4(4x^1+1))^2)$				
FB	GP	0.025 ± 0.023	1559.6 ± 126	400.0
ABSUR	GP	0.027 ± 0.028	90.7 ± 11.8	52.1
RB	GP	0.025 ± 0.035	61.5 ± 7.5	51.0
MLB	GP	0.026 ± 0.035	70.8 ± 13.1	54.1
ADSA	GP	0.031 ± 0.035	112.9 ± 6.4	45.2
DDSA	GP	0.026 ± 0.031	66.2 ± 4.3	52.0
FB	t-GP	0.020 ± 0.007	2966.6 ± 604	400.0
ABSUR	t-GP	0.023 ± 0.008	171.7 ± 34.0	47.6
RB	t-GP	0.022 ± 0.009	126.7 ± 14.5	50.8
MLB	t-GP	0.023 ± 0.008	129.7 ± 21.0	51.4
ADSA	t-GP	0.025 ± 0.028	222.0 ± 23.2	36.5
DDSA	t-GP	0.025 ± 0.008	133.7 ± 7.5	52.0
6D Hartman WITH $\epsilon \sim \mathcal{N}(0,1)$ AND $N_T = 6000$				
FB	GP	0.030 ± 0.004	1934 ± 74	600
ABSUR	GP	0.070 ± 0.015	289 ± 40	160
RB	GP	0.058 ± 0.014	105 ± 34	143
MLB	GP	0.037 ± 0.008	294 ± 133	241
ADSA	GP	0.043 ± 0.007	199 ± 38	172
DDSA	GP	0.050 ± 0.009	101 ± 16	142
6D Hartman WITH $\epsilon \sim \mathcal{N}(0,1)$ AND $N_T = 30000$				
FB $r_n = 50$	GP	0.015 ± 0.002	1654 ± 347.0	600.0
FB $r_n = 100$	GP	0.016 ± 0.002	461.5 ± 18.61	330.0
FB $r_n = 200$	GP	0.029 ± 0.006	152.2 ± 6.693	195.0
ABSUR	GP	0.022 ± 0.003	757.2 ± 34.77	325.3
RB	GP	0.024 ± 0.005	227.0 ± 64.04	237.0
MLB	GP	0.022 ± 0.006	240.6 ± 76.91	242.9
ADSA	GP	0.016 ± 0.002	995.6 ± 52.65	373.8
DDSA	GP	0.017 ± 0.002	522.0 ± 26.17	350.0

Table 5.3: Parameter set-ups for Bermudan Option examples.

Scheme	2-D Basket Put	3-D Max-Call
–	$N_T = 2000$; $k_0 = 20, r_0 = 20$	$N_T = 30,000$; $k_0 = 300, r_0 = 30$
–		$N_T = 4,500$; $k_0 = 30, r_0 = 30$
FB	$r = 20$	$r = 30$
MLB/RB	$\mathbf{r}_L = \{20, 30, 40, 50,$ $60, 80, 120, 160\}$	$\mathbf{r}_L = \{20, 30, 40, 50, 80,$ $160, 240, 320, 480, 640\}$
ABSUR	$\mathcal{R} = [20, 160], T_{sim} = 0.01$	$\mathcal{R} = [20, 640], T_{sim} = 0.01$

Table 5.4: Performance of GP metamodels with FB, MLB, RB, ABSUR, ADSA and DDSA designs in the 2-D Average Put and 3-D Max Call examples. Results are averages from 20 runs of each scheme.

DESIGN	MODEL	PAYOFF	TIME/S T	INPUTS k_T
2-D AVERAGE PUT				
FB	GP	1.451± 0.002	68.87	100.00
RB	GP	1.447± 0.003	27.93	51.96
MLB	GP	1.448± 0.003	26.37	55.42
ABSUR	GP	1.448± 0.003	23.52	55.36
ADSA	GP	1.447 ± 0.002	24.20	42.57
DDSA	GP	1.441 ± 0.006	17.58	35.00
FB	t-GP	1.448 ± 0.002	74.57	100.00
RB	t-GP	1.446 ± 0.002	60.55	54.59
MLB	t-GP	1.448 ± 0.003	35.00	57.40
ABSUR	t-GP	1.441 ± 0.005	46.00	51.48
ADSA	t-GP	1.448 ± 0.002	55.79	47.99
DDSA	t-GP	1.446 ± 0.003	22.35	35.00
3-D MAX CALL $N_T = 4500$				
FB	GP	11.15 ± 0.018	296.6	150.0
RB	GP	11.13 ± 0.015	135.6	91.5
MLB	GP	11.13 ± 0.019	135.9	91.1
ABSUR	GP	11.07 ± 0.019	104.7	79.4
ADSA	GP	11.14 ± 0.017	168.7	98.9
DDSA	GP	11.11 ± 0.022	157.5	65.0
3-D MAX CALL $N_T = 30000$				
FB	GP	11.26 ± 0.010	2425.0	1000.0
RB	GP	11.25 ± 0.008	410.0	562.5
MLB	GP	11.25 ± 0.008	520.3	608.2
ABSUR	GP	11.23 ± 0.007	122.1	408.6
ADSA	GP	11.25 ± 0.009	223.7	461.8
DDSA	GP	11.24 ± 0.009	239.4	385.0

Chapter 6

Conclusions and Future Works

In this thesis, we mainly study the problem of noisy level set estimation of expensive black-box functions. These problems arise in industrial context, as an example, we learn experimentation on optimal stopping for Bermudan Option.

Existing works on noisy level set estimation mainly focus on developing SUR-related sequential design frameworks for Gaussian Process metamodels. However, in real application, it is hard to satisfy the Gaussian assumption on noise and thus not appropriate to use Gaussian observation GP. Instead, one of the main contributions of this thesis aims at providing alternative Gaussian Process related surrogates for misspecified and heteroscedastic noise or low signal to noise ratio settings. In Chapter 3 and 4, we have carried a comprehensive comparison of five metamodels and four design heuristics on 18 case studies (4×3 synthetic, plus six real-world). In sum, the considered alternatives to standard Gaussian-observation GP do perform somewhat better. In particular, t-GP directly nests plain GP and hence essentially always matches or exceeds the performance of the latter. We also observe gains from using Cl-GP when SNR is low and from monotonic surrogates when the underlying response is monotone. That being said, final recommendation regarding the associated benefit depends on computational considerations, as the respective overhead becomes larger (and exact updating of the metamodel no longer

possible).

In terms of design, we advocate the benefits of tMSE, which generates high-performing experimental designs without requiring expensive acquisition function (or even look-ahead variance). The tMSE criterion does sometimes suffer from the tendency to put many designs at the edge of the input space but otherwise tends to match the performance of more complex and computationally intensive $\mathcal{I}_n$'s. For expensive simulations, ICU is probably still the best choice (although in that case, random-set-based heuristics should also be considered). Especially in higher dimensions with misspecified noise, ICU is the best choice among all designs. We also stress that the user ought to thoughtfully pick the *combination* of sequential design and metamodel, since cross-dependencies are involved (e.g.,classification metamodels generally not working well with the ICU criterion in lower dimension).

To deal with mis-specified noise or low signal to noise ratio setting, besides using a more robust metamodel to construct the surrogates for the underlying function, another more intuitive approach is to increase the design size, or the information we obtain. However, a large number of total simulations N_T has been identified as one of the main challenges for surrogate based methods in terms of both computational efficiency and memorization capacity. One potential break-through has been performed with batching/replication algorithm, using the same input to generate multiple responses, and it has been investigated for mean response modeling problem [22, 9, 10]. Batching also helps to increase the signal to noise ratio at individual input and thus stands out in stochastic experiments where noise is large or misspecified. In Chaper 5, we have proposed and investigated five different schemes for adaptive batching in metamodeling of stochastic experiments in level set estimation. All schemes successfully capture the intuition of increasingly beneficial replication as sequential design is constructed and the focus shifts from exploration to exploitation. Our algorithms are based on the plain Gaussian Process paradigm but are easily extended to related non-Gaussian frameworks, as demonstrated with t-GP. The key step is to construct an approximation of the batch look-ahead

variance $s^{(n+1)}(x, r)$. Our results demonstrate that adaptive batching offers a simple mechanism to extract significant computational gains through building more compact designs and taking advantage of the symbiotic relationship between GPs and replication. Thus, compared with using a constant value for replicates r over all inputs like in FB, we are able to gain more than an order-of-magnitude speed-up with minimal loss of metamodeling fidelity with adaptive batching designs for noisy level set estimation problems. Among all five schemes for adaptive batching, we recommend using ADSA, which shines out especially in higher dimension or complicated setups. Although its simplified version DDSA is more efficient and performs as well as ADSA in lower dimension, it turns out to be less accurate than ADSA in more complicated problems. RB and MLB are super efficient. However, they sometimes suffer from less accuracy due to being over aggressive. Performance of ABSUR largely depends on tuning the parameters in the scheme, and cSUR strategy converges in lower rate in higher dimensions.

Our focus of this thesis has been on alternative GP metamodels and sequential design in the context of level-set estimation. Related problems such as evaluating the probability of failure, or evaluating a tail risk measure, would benefit from the same ideas and will be investigated in follow-up projects. More widely, the same idea can be applied in optimization and surface metamodeling problems, with further investigation on the corresponding acquisition function. Our work has provided efficient computational recipe of look-ahead variance for all five alternative metamodels, which can be shared among the similar problems. Besides the metamodel, recent publications also propose approaches for robust estimate of parameters in GP [40, 26] and show improved performance in surrogate modeling. Extensions of Gaussian Processes in Deep Learning is another current trend, with topics related with Deep Gaussian Process (DGP) in surface metamodeling [87] and optimization [41], that replace the Gaussian

Processes we used in our work. In Deep Gaussian Process, the observation y is modeled via

$$y(x) = f_L(\mathbf{f}_{L-1}(...(\mathbf{f}_1(\mathbf{f}_0(x))))) + \epsilon(x) \tag{6.1}$$

where L is the number of layers and $\mathbf{f}_l(\cdot)$ is an intermediate GP. Since DGP is a functional composition of GPs, it was also named as a hierarchical GP [29]. It has been shown that DGPs handle scarce data and overcome the overfitting issue which is common in optimization problem involving expensive black-box functions and induced uncertainty produced while dealing with scarce data [41]. However, literature on inference with DGP, including hyper-parameter selection, and derivatation of sequential design strategies for DGP still remains incomplete. Acceleration of GP computation with advanced machine is discussed in [68, 36], which makes deep Gaussian Process and GP for high-dimensional optimization and level set estimation practical in real applications.

Another important problem that is beyond the scope of the present work is theoretical analysis about the asymptotic complexity of the proposed schemes such as ADSA, for example to establish the long-run growth rate of k_n in order to quantify the asymptotic complexity as $N_n \to \infty$. Several recent publications [18, 3] focus on asymptotic analysis of Gaussian Process. However, research on asymptotic complexity of adaptive batching design or even sequential design still remains open.

Bibliography

[1] B. Ankenman, B. L. Nelson, and J. Staum, *Stochastic kriging for simulation metamodeling*, *Operations research* **58** (2010), no. 2 371–382.

[2] D. Azzimonti, J. Bect, C. Chevalier, and D. Ginsbourger, *Quantifying uncertainties on excursion sets under a Gaussian random field prior*, *SIAM/ASA Journal on Uncertainty Quantification* **4** (2016), no. 1 850–874.

[3] F. Bachoc *et. al.*, *Asymptotic analysis of covariance parameter estimation for gaussian processes in the misspecified case*, *Bernoulli* **24** (2018), no. 2 1531–1575.

[4] V. Bally, G. Pagès, and J. Printems, *A quantization tree method for pricing and hedging multidimensional american options*, *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics* **15** (2005), no. 1 119–168.

[5] M. J. Bayarri, J. O. Berger, E. S. Calder, K. Dalbey, S. Lunagomez, A. K. Patra, E. B. Pitman, E. T. Spiller, and R. L. Wolpert, *Using statistical and computer models to quantify volcanic hazards*, *Technometrics* **51** (2009), no. 4 402–413.

[6] J. Bect, D. Ginsbourger, L. Li, V. Picheny, and E. Vazquez, *Sequential design of computer experiments for the estimation of a probability of failure*, *Statistics and Computing* **22** (2012), no. 3 773–793.

[7] B. J. Bichon, M. S. Eldred, L. P. Swiler, S. Mahadevan, and J. M. McFarland, *Efficient global reliability analysis for nonlinear implicit performance functions*, *AIAA Journal* **46** (2008), no. 10 2459–2468.

[8] M. Binois, R. B. Gramacy, and M. Ludkovski, *Practical heteroskedastic Gaussian process modeling for large simulation experiments*, *Journal of Computational and Graphical Statistics* **0** (2018), no. ja 1–41, [https://doi.org/10.1080/10618600.2018.1458625].

[9] M. Binois, J. Huang, R. B. Gramacy, and M. Ludkovski, *Replication or exploration? Sequential design for stochastic simulation experiments*, *Technometrics* **0** (2018), no. ja 1–43, [https://doi.org/10.1080/00401706.2018.1469433].

[10] M. Binois, J. Huang, R. B. Gramacy, and M. Ludkovski, *Replication or exploration? Sequential design for stochastic simulation experiments*, *Technometrics* (2018), no. just-accepted 1–43.

[11] I. Bogunovic, J. Scarlett, A. Krause, and V. Cevher, *Truncated variance reduction: A unified approach to bayesian optimization and level-set estimation*, in *Advances in neural information processing systems*, pp. 1507–1515, 2016.

[12] B. Bouchard, I. Ekeland, and N. Touzi, *On the malliavin approach to monte carlo approximation of conditional expectations*, *Finance and Stochastics* **8** (2004), no. 1 45–71.

[13] A. Boukouvalas and D. Cornford, *Learning heteroscedastic gaussian processes for complex datasets*, *Technical report* (2009).

[14] K. M. Bretthauer, A. Ross, and B. Shetty, *Nonlinear integer programming for optimal allocation in stratified sampling*, *European Journal of Operational Research* **116** (1999), no. 3 667–680.

[15] M. Broadie, P. Glasserman, *et. al.*, *A stochastic mesh method for pricing high-dimensional american options*, *Journal of Computational Finance* **7** (2004) 35–72.

[16] B. Bryan, R. C. Nichol, C. R. Genovese, J. Schneider, C. J. Miller, and L. Wasserman, *Active learning for identifying function threshold boundaries*, in *Advances in neural information processing systems*, pp. 163–170, 2006.

[17] B. Bryan and J. Schneider, *Actively learning level-sets of composite functions*, in *Proceedings of the 25th International Conference on Machine Learning*, pp. 80–87, ACM, 2008.

[18] D. R. Burt, C. E. Rasmussen, and M. Van Der Wilk, *Rates of convergence for sparse variational gaussian process regression*, *arXiv preprint arXiv:1903.03571* (2019).

[19] J. F. Carriere, *Valuation of the early-exercise price for options using simulations and nonparametric regression*, *Insurance: mathematics and Economics* **19** (1996), no. 1 19–30.

[20] C.-H. Chen, J. Lin, E. Yücesan, and S. E. Chick, *Simulation budget allocation for further enhancing the efficiency of ordinal optimization*, *Discrete Event Dynamic Systems* **10** (2000), no. 3 251–270.

[21] X. Chen and K.-K. Kim, *Building metamodels for quantile-based measures using sectioning*, in *2013 Winter Simulations Conference (WSC)*, pp. 521–532, IEEE, 2013.

[22] X. Chen and Q. Zhou, *Sequential design strategies for mean response surface metamodeling via stochastic kriging with adaptive exploration and exploitation*, *European Journal of Operational Research* **262** (2017), no. 2 575–585.

[23] C. Chevalier, J. Bect, D. Ginsbourger, E. Vazquez, V. Picheny, and Y. Richet, *Fast parallel kriging-based stepwise uncertainty reduction with application to the identification of an excursion set*, *Technometrics* **56** (2014), no. 4 455–465.

[24] C. Chevalier, D. Ginsbourger, J. Bect, and I. Molchanov, *Estimating and quantifying uncertainties on level sets using the Vorob'ev expectation and deviation with Gaussian process models*, in *mODa 10–Advances in Model-Oriented Design and Analysis*, pp. 35–43. Springer, 2013.

[25] C. Chevalier, D. Ginsbourger, and X. Emery, *Corrected kriging update formulae for batch-sequential data assimilation*, in *Mathematics of Planet Earth*, pp. 119–122. Springer, 2014.

[26] M. Chung, M. Binois, R. B. Gramacy, J. M. Bardsley, D. J. Moquin, A. P. Smith, and A. M. Smith, *Parameter and uncertainty estimation for dynamical systems using surrogate stochastic processes*, *SIAM Journal on Scientific Computing* **41** (2019), no. 4 A2212–A2238.

[27] C. Cioffi-Revilla, *Introduction to computational social science*, *London and Heidelberg: Springer* (2014).

[28] M. Cutler, T. J. Walsh, and J. P. How, *Reinforcement learning with multi-fidelity simulators*, in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3888–3895, IEEE, 2014.

[29] A. Damianou and N. Lawrence, *Deep gaussian processes*, in *Artificial Intelligence and Statistics*, pp. 207–215, 2013.

[30] V. Dubourg, B. Sudret, and J.-M. Bourinet, *Reliability-based design optimization using kriging surrogates and subset simulation*, *Structural and Multidisciplinary Optimization* **44** (2011), no. 5 673–690.

[31] B. Echard, N. Gayton, and M. Lemaire, *Kriging based Monte Carlo simulation to compute the probability of failure efficiently: AK-MCS method*, *6emes Journées Nationales de Fiabilité, 24–26 mars, Toulouse, France* (2010).

[32] B. Echard, N. Gayton, and M. Lemaire, *Ak-mcs: an active learning reliability method combining kriging and monte carlo simulation*, *Structural Safety* **33** (2011), no. 2 145–154.

[33] K.-T. Fang, D. K. Lin, P. Winker, and Y. Zhang, *Uniform design: theory and application*, *Technometrics* **42** (2000), no. 3 237–248.

[34] P. I. Frazier, W. B. Powell, and S. Dayanik, *A knowledge-gradient policy for sequential information collection*, *SIAM Journal on Control and Optimization* **47** (2008), no. 5 2410–2439.

[35] J. P. French, S. R. Sain, *et. al.*, *Spatio-temporal exceedance locations and confidence regions*, *The Annals of Applied Statistics* **7** (2013), no. 3 1421–1449.

[36] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, *Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration*, in *Advances in Neural Information Processing Systems*, pp. 7576–7586, 2018.

[37] J. Gonzalez, J. Longworth, D. C. James, and N. D. Lawrence, *Bayesian optimization for synthetic gene design*, *arXiv preprint arXiv:1505.01627* (2015).

[38] A. Gotovos, N. Casati, G. Hitz, and A. Krause, *Active learning for level set estimation*, in *IJCAI*, 2013.

[39] R. B. Gramacy and M. Ludkovski, *Sequential design for optimal stopping problems*, *SIAM Journal on Financial Mathematics* **6** (2015), no. 1 748–775.

[40] M. Gu, X. Wang, J. O. Berger, *et. al.*, *Robust gaussian stochastic process emulation*, *The Annals of Statistics* **46** (2018), no. 6A 3038–3066.

[41] A. Hebbal, L. Brevault, M. Balesdent, E.-G. Taibi, and N. Melab, *Efficient global optimization using deep gaussian processes*, in *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, IEEE, 2018.

[42] P. Hennig and C. J. Schuler, *Entropy search for information-efficient global optimization*, *Journal of Machine Learning Research* **13** (2012), no. Jun 1809–1837.

[43] L. J. Hong and B. L. Nelson, *Discrete optimization via simulation using compass*, *Operations Research* **54** (2006), no. 1 115–129.

[44] R. Hu and M. Ludkovski, *Sequential design for ranking response surfaces*, *SIAM/ASA Journal on Uncertainty Quantification* **5** (2017), no. 1 212–239.

[45] D. Huang, T. T. Allen, W. I. Notz, and N. Zeng, *Global optimization of stochastic black-box systems via sequential kriging meta-models*, *Journal of global optimization* **34** (2006), no. 3 441–466.

[46] F. Hutter, H. H. Hoos, and K. Leyton-Brown, *Sequential model-based optimization for general algorithm configuration*, in *International conference on learning and intelligent optimization*, pp. 507–523, Springer, 2011.

[47] H. Jalali, I. Van Nieuwenhuyse, and V. Picheny, *Comparison of kriging-based algorithms for simulation optimization with heterogeneous noise*, *European Journal of Operational Research* **261** (2017), no. 1 279–301.

[48] H. Jalali, I. Van Nieuwenhuyse, and V. Picheny, *Comparison of Kriging-based methods for simulation optimization with heterogeneous noise*, *European Journal of Operational Research* **261** (2017), no. 1 279–301.

[49] L. R. Johnson, *Microcolony and biofilm formation as a survival strategy for bacteria*, *Journal of Theoretical Biology* **251** (2008), no. 1 24–34.

[50] M. E. Johnson, L. M. Moore, and D. Ylvisaker, *Minimax and maximin distance designs*, *Journal of statistical planning and inference* **26** (1990), no. 2 131–148.

[51] D. R. Jones, M. Schonlau, and W. J. Welch, *Efficient global optimization of expensive black-box functions*, *Journal of Global Optimization* **13** (1998), no. 4 455–492.

[52] P. Jylänki, J. Vanhatalo, and A. Vehtari, *Robust Gaussian process regression with a Student-t likelihood*, *Journal of Machine Learning Research* **12** (2011), no. Nov 3227–3257.

[53] K. Kandasamy, G. Dasarathy, J. B. Oliva, J. Schneider, and B. Póczos, *Gaussian process bandit optimisation with multi-fidelity evaluations*, in *Advances in Neural Information Processing Systems*, pp. 992–1000, 2016.

[54] K. Kandasamy, G. Dasarathy, B. Poczos, and J. Schneider, *The multi-fidelity multi-armed bandit*, in *Advances in Neural Information Processing Systems*, pp. 1777–1785, 2016.

[55] K. Kandasamy, G. Dasarathy, J. Schneider, and B. Poczos, *Multi-fidelity Bayesian optimisation with continuous approximations*, *arXiv preprint arXiv:1703.06240* (2017).

[56] J. P. Kleijnen, *Design and analysis of simulation experiments*, in *International Workshop on Simulation*, pp. 3–22, Springer, 2015.

[57] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, *Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets*, in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (A. Singh and J. Zhu, eds.), vol. 54 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 528–536, PMLR, 20–22 Apr, 2017.

[58] J. Koehler, A. Puhalskii, and B. Simon, *Estimating functions evaluated by simulation: A Bayesian/analytic approach*, *Annals of Applied Probability* **8** (1998), no. 4 1184–1215.

[59] L. Le Gratiet and J. Garnier, *Asymptotic analysis of the learning curve for Gaussian process regression*, *Journal of Machine Learning Research* **98** (2015), no. 3 407–433.

[60] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, *Hyperband: A novel bandit-based approach to hyperparameter optimization*, *arXiv preprint arXiv:1603.06560* (2016).

[61] M. Liu and J. Staum, *Stochastic kriging for efficient nested simulation of expected shortfall*, *Journal of Risk* **12** (2010), no. 3 3–27.

[62] D. J. Lizotte, T. Wang, M. H. Bowling, and D. Schuurmans, *Automatic gait optimization with gaussian process regression.*, in *IJCAI*, vol. 7, pp. 944–949, 2007.

[63] F. A. Longstaff and E. S. Schwartz, *Valuing American options by simulation: a simple least-squares approach*, *Review of Financial Studies* **14** (2001), no. 1 113–147.

[64] M. Ludkovski, *Kriging metamodels and experimental design for Bermudan option pricing*, *Journal of Computational Finance* **22** (2018), no. 1 37–77. arXiv preprint arXiv:1509.02179.

[65] M. Ludkovski and J. Niemi, *Optimal dynamic policies for influenza management*, *Statistical Communications in Infectious Diseases* **2** (2010), no. 1.

[66] M. Ludkovski and J. Risk, *Sequential design and spatial modeling for portfolio tail risk measurement*, *SIAM Journal on Financial Mathematics* **9** (2018), no. 4 1137–1174.

[67] X. Lyu, M. Binois, and M. Ludkovski, *Evaluating Gaussian process metamodels and sequential designs for noisy level set estimation*, *arXiv preprint arXiv:1807.06712* (2018).

[68] G. Manogaran and D. Lopez, *A gaussian process based big data processing framework in cluster computing environment*, *Cluster Computing* **21** (2018), no. 1 189–204.

[69] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos, *Active policy learning for robot planning and exploration under uncertainty.*, in *Robotics: Science and Systems*, vol. 3, pp. 321–328, 2007.

[70] M. D. McKay, R. J. Beckman, and W. J. Conover, *Comparison of three methods for selecting values of input variables in the analysis of output from a computer code*, *Technometrics* **21** (1979), no. 2 239–245.

[71] M. McLeod, M. A. Osborne, and S. J. Roberts, *Practical Bayesian optimization for variable cost objectives*, *arXiv preprint arXiv:1703.04335* (2017).

[72] E. Mehdad and J. P. Kleijnen, *Stochastic intrinsic kriging for simulation metamodeling*, *Applied Stochastic Models in Business and Industry* **34** (2018), no. 3 322–337.

[73] T. P. Minka, *Expectation propagation for approximate Bayesian inference*, in *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pp. 362–369, Morgan Kaufmann Publishers Inc., 2001.

[74] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, *Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS*, *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* **24** (2005), no. 12 1859–1880.

[75] B. L. Nelson, *Stochastic Modeling: Analysis & Simulation*. Courier Corporation, 2010.

[76] S. H. Ng and J. Yin, *Bayesian kriging analysis and design for stochastic simulations*, *ACM Transactions on Modeling and Computer Simulation (TOMACS)* **22** (2012), no. 3 17.

[77] A. O'Hagan, *On outlier rejection phenomena in Bayes inference*, *Journal of the Royal Statistical Society. Series B (Methodological)* (1979) 358–367.

[78] C. Osorio and M. Bierlaire, *A simulation-based optimization framework for urban transportation problems*, *Operations Research* **61** (2013), no. 6 1333–1345.

[79] A. B. Owen, *Orthogonal arrays for computer experiments, integration and visualization*, *Statistica Sinica* (1992) 439–452.

[80] D. Parkinson, P. Mukherjee, and A. R. Liddle, *Bayesian model selection analysis of wmap3*, *Physical review D* **73** (2006), no. 12 123523.

[81] V. Picheny and D. Ginsbourger, *A nonstationary space-time Gaussian process model for partially converged simulations*, *SIAM/ASA Journal on Uncertainty Quantification* **1** (2013), no. 1 57–78.

[82] V. Picheny, D. Ginsbourger, Y. Richet, and G. Caplin, *Quantile-based optimization of noisy computer experiments with tunable precision*, *Technometrics* **55** (2013), no. 1 2–13.

[83] V. Picheny, D. Ginsbourger, O. Roustant, R. T. Haftka, and N.-H. Kim, *Adaptive designs of experiments for accurate approximation of a target region*, *Journal of Mechanical Design* **132** (2010), no. 7 071008.

[84] V. Picheny, T. Wagner, and D. Ginsbourger, *A benchmark of kriging-based infill criteria for noisy optimization*, *Structural and Multidisciplinary Optimization* **48** (2013), no. 3 607–626.

[85] M. Poloczek, J. Wang, and P. Frazier, *Multi-information source optimization*, in *Advances in Neural Information Processing Systems*, pp. 4288–4298, 2017.

[86] N. Quan, J. Yin, S. H. Ng, and L. H. Lee, *Simulation optimization via kriging: a sequential search using expected improvement with computing budget constraints*, *Iie Transactions* **45** (2013), no. 7 763–780.

[87] M. I. Radaideh and T. Kozlowski, *Surrogate modeling of advanced computer simulations using deep gaussian processes*, *Reliability Engineering & System Safety* **195** (2020) 106731.

[88] P. Ranjan, D. Bingham, and G. Michailidis, *Sequential experiment design for contour estimation from complex computer codes*, *Technometrics* **50** (2008), no. 4 527–541.

[89] P. Ranjan, D. Bingham, and G. Michailidis, *Sequential experiment design for contour estimation from complex computer codes*, *Technometrics* **50** (2008), no. 4 527–541.

[90] C. E. Rasmussen and H. Nickisch, *Gaussian processes for machine learning (gpml) toolbox*, *Journal of Machine Learning Research* **11** (2010), no. Nov 3011–3015.

[91] J. Riihimäki and A. Vehtari, *Gaussian processes with monotonicity information*, in *AISTATS*, vol. 9, pp. 645–652, 2010.

[92] T. J. Robinson, J. B. Birch, and B. A. Starnes, *A semi-parametric approach to dual modeling when no replication exists*, *Journal of Statistical Planning and Inference* **140** (2010), no. 10 2860–2869.

[93] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*. Springer Science & Business Media, 2013.

[94] A. Shah, A. Wilson, and Z. Ghahramani, *Student-t processes as alternatives to Gaussian processes*, in *Artificial Intelligence and Statistics*, pp. 877–885, 2014.

[95] M. C. Shewry and H. P. Wynn, *Maximum entropy sampling*, *Journal of applied statistics* **14** (1987), no. 2 165–170.

[96] J. Snoek, H. Larochelle, and R. P. Adams, *Practical bayesian optimization of machine learning algorithms*, in *Advances in neural information processing systems*, pp. 2951–2959, 2012.

[97] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, *Information-theoretic regret bounds for Gaussian process optimization in the bandit setting*, *IEEE Transactions on Information Theory* **58** (2012), no. 5 3250–3265.

[98] J. Staum, *Better simulation metamodeling: The why, what, and how of stochastic kriging*, in *Proceedings of the 2009 Winter Simulation Conference (WSC)*, pp. 119–133, IEEE, 2009.

[99] R. Stroh, S. Demeyer, N. Fischer, J. Bect, and E. Vazquez, *Sequential design of experiments to estimate a probability of exceeding a threshold in a multi-fidelity stochastic simulator*, *arXiv preprint arXiv:1707.08384* (2017).

[100] K. Swersky, J. Snoek, and R. P. Adams, *Multi-task Bayesian optimization*, in *Advances in neural information processing systems*, pp. 2004–2012, 2013.

[101] B. Tang, *Orthogonal array-based latin hypercubes*, *Journal of the American statistical association* **88** (1993), no. 424 1392–1397.

[102] J. N. Tsitsiklis and B. Van Roy, *Regression methods for pricing complex American-style options*, *IEEE Transactions on Neural Networks* **12** (2001), no. 4 694–703.

[103] W. C. Van Beers and J. P. Kleijnen, *Customized sequential designs for random simulation experiments: Kriging metamodeling and bootstrapping*, *European journal of operational research* **186** (2008), no. 3 1099–1113.

[104] J. Vanhatalo, J. Riihimäki, J. Hartikainen, P. Jylänki, V. Tolvanen, and A. Vehtari, *Bayesian modeling with Gaussian processes using the GPstuff toolbox*, *The Journal of Machine Learning Research* **14** (2013), no. 1 1175–1179.

[105] J. Vanhatalo, J. Riihimäki, J. Hartikainen, P. Jylänki, V. Tolvanen, and A. Vehtari, *GPstuff: Bayesian modeling with Gaussian processes*, *Journal of Machine Learning Research* **14** (2013), no. Apr 1175–1179.

[106] J. Villemonteix, E. Vazquez, and E. Walter, *An informational approach to the global optimization of expensive-to-evaluate functions*, *Journal of Global Optimization* **44** (2009), no. 4 509.

[107] Z. Wang, J. Q. Shi, and Y. Lee, *Extended t-process regression models*, *Journal of Statistical Planning and Inference* **189** (2017) 38–60.

[108] C. K. Williams and D. Barber, *Bayesian classification with Gaussian processes*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20** (1998), no. 12 1342–1351.

[109] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[110] F. Yang, B. E. Ankenman, and B. L. Nelson, *Estimating cycle time percentile curves for manufacturing systems via simulation*, *INFORMS Journal on Computing* **20** (2008), no. 4 628–643.